P. POPOVICI
O. CIRA

REZOLVAREA NUMERICĂ A ECUAȚIILOR NELINIARE

SIGNATA
SRL

Scanned with OKEN Scanner

Dr. PARASCHIVA POPOVICI
lector universitar

OCTAVIAN CIRA
lector universitar

REZOLVAREA NUMERICĂ A ECUAŢIILOR NELINIARE

Scanned with OKEN Scanner

ISBN 973-551-0025-1

Bun de tipar: 15.08.1991
Apărut: 1992
Coli tipar: 17

Scanned with OKEN Scanner

Dr. PARASCHIVA POPOVICI
lector universitar

OCTAVIAN CIRA
lector universitar

REZOLVAREA NUMERICĂ A ECUAȚIILOR NELINIARE

EDITURA SIGNATA TIMIȘOARA
S.R.L.
1992

Scanned with OKEN Scanner

Specialişti ai diverselor domenii de activitate au fost, sînt şi vor fi permanent confruntaţi cu rezolvarea ecuaţiilor operatoriale ($Fx = b$, $F : X \to Y$, X, Y spaţii finit dimensionale sau spaţii Hilbert reale...) de o diversitate de tipuri.

În general, nu este posibil de a determina explicit soluţia ecuaţiei ($Fx = b$) sau, dacă se poate, forma ei explicită este atît de complicată încît ea este inutilizabilă şi din acest motiv căutăm o soluţie aproximativă.

În cadrul aproximării soluţiilor ecuaţiilor, studiul metodelor iterative ocupă un loc important.

În cartea de faţă sînt tratate metode numerice iterative proprii sau generalizări ale unor metode iterative cunoscute.

Capitolul 1 cuprinde metode de rezolvare a ecuaţiilor algebrice (Metode de determinare a zerourilor polinoamelor algebrice, Metode de tip Lehmer-Schur) şi programele FORTRAN 77 corespunzătoare.

În Capitolele 2 şi 3 se abordează rezolvarea numerică a ecuaţiilor neliniare în spaţii finit dimensionale sau spaţii Hilbert reale, ecuaţii care provin prin discretizarea unor ecuaţii diferenţiale neliniare, în mod special a ecuaţiilor cu diferenţe corespunzătoare problemelor la limită bilocale pentru ecuaţii diferenţiale slab neliniare, sau cvasiliniare, problemei Dirichlet pentru ecuaţii eliptice slab neliniare. Întrucît ecuaţiile cu diferenţe sînt în general lacunare, derivata Fréchet fiind o matrice rară, metodele numerice studiate în Capitolul 3 (Metode itera-

Scanned with OKEN Scanner

ţiilor simple, Metoda direcţiilor alternate, Metode de tip gradient, O metodă iterativă specială pentru ecuaţii de forma (2.12), Metoda lui Fridman) se referă la ecuaţii neliniare de acest tip. Metodele de calcul cu matrice rare, pentru a fi eficiente, trebuie să beneficieze de proporţia mare de elemente nule din aceste matrice, ceea ce creează necesitatea considerării unor tehnici speciale de stocare şi prelucrare, în vederea obţinerii unor programe calculator cu consum redus de memorie internă şi timp unitate centrală.

În Capitolul 4, în care sînt descrişi algoritmii de calcul şi programele FORTRAN corespunzătoare metodelor prezentate în Capitolul 3, s-au studiat diverse tehnici de memorare, între care memorarea compactă aleatoare ocupă un loc central.

În totalitatea lor cele patru capitole realizează o tratare completă a subiectului, de la fundamentări teoretice pînă la procedee numerice şi programe calculator.

Prin conţinutul ei lucrarea este adresată inginerilor, matematicienilor, informaticienilor, studenţilor, precum şi celor care în activitatea lor practică au de rezolvat probleme pe calculator.

Autorii

CUPRINS

1. Rezolvarea numerică a ecuaţiilor algebrice 7

1.1. Metode de determinare a zerourilor polinoamelor algebrice 7

1.2. Mărginea zerourilor polinoamelor algebrice 8

1.3. Determinarea simultană a zerourilor reale ale polinoamelor din $\mathbb{R}[x]$ 29

1.4. Metode de tip Lehmer-Schur pentru determinarea zerourilor polinoamelor din $\mathbb{C}[x]$ 44

1.5. Metode cu ordine superioară de convergenţă de determinare simultană a zerourilor polinoamelor algebrice 74

1.6. Metode de determinare a zerourilor multiple 89

2. Ecuaţii cu diferenţe corespunzătoare problemelor la limită neliniare 93

2.1. Problema la limită bilocală pentru ecuaţii diferenţiale slab neliniare 93

2.1.1. Notaţii şi noţiuni introductive 94

2.1.2. Problema la limită bilocală slab neliniară 96

2.1.3. Ecuaţii cu diferenţe finite corespunzătoare problemei (2.2) 97

2.2. Problema la limită bilocală pentru ecuaţii diferenţiale cvasiliniare 100

2.2.1. Noţiuni introductive 100

2.2.2. Probleme la limită bilocală cvasiliniară şi ecuaţii cu diferenţe finite standard 101
2.2.3. Caz particular 103
2.3. Problema Dirichlet pentru ecuaţii eliptice slab neliniare 104
2.3.1. Notaţii şi noţiuni introductive 104
2.3.2. Diferenţe finite 106
2.3.3. Ecuaţii cu diferenţe finite standard 109

3. Convergenţa locală a unor metode iterative 115
3.1. Introducere 116
3.1.1. Teme preliminare 116
3.1.2. Convergenţa locală 117
3.1.3. Operatori cvasi-necxpensivi 118
3.1.4. Operatori monotoni 120
3.2. Determinarea iteraţiei de start pentru metode iterative 121
3.2.1. Căutarea iteraţiei de start în nodurile unei reţele de puncte 121
3.2.2. Căutarea aleatoare a iteraţiei de start 122
3.2.3. Căutarea iteraţiei de start cu ajutorul gradului topologic 122
3.3. Inversarea matricilor celulor tridiagonale şi simetrice 149
3.4. Metoda iteraţiilor simple pentru ecuaţii de forma (2.12) 157
3.4.1. Algoritmul de calcul şi analiza convergenţei . . . 160
3.4.2. Aplicaţie la rezolvarea ecuaţiilor cu diferenţe corespunzătoare problemelor la limită bilocale cvasiliniare 170

Scanned with OKEN Scanner

3.4.3. Rezolvarea problemelor la limită slab neliniare cu ajutorul ecuaţiilor cu diferenţe finite corectate 174
3.5. Metoda direcţiilor alternate 178
3.5.1. Introducere 178
3.5.2. Leme preliminare 179
3.5.3. Analiza convergenţei 182
3.6. Metode de tip gradient 183
3.6.1. Introducere 183
3.6.2. Analiza convergenţei 184
3.6.3. Metoda gradientului 186
3.6.4. Metoda Newton - gradient combinată 187
3.6.5. Metoda lui Fridman 189
3.6.6. Observaţii asupra metodei gradientului generalizată cu ordin superior de convergenţă 190
3.7. O metodă iterativă specială pentru ecuaţii de forma (2.12) 191
3.7.1. Algoritmul de calcul şi analiza convergenţei . . . 191
3.7.2. Metoda liniilor paralele 194
3.7.3. Aplicaţie la ecuaţiile cu diferenţe corespunzătoare problemei Dirichlet eliptice slab neliniare . . 195
3.8. Metoda lui Fridman 198
3.8.1. Introducere 198
3.8.2. Analiza convergenţei 199

4. Algoritmi de calcul şi programe 201
4.1. Metoda iteraţiilor simple 201
4.1.1. Algoritmul de calcul 201
4.1.2. Cazul ecuaţiilor cu diferenţe (2.4) 205
4.1.3. Cazul ecuaţiilor cu diferenţe cu corecţii (2.5) şi (2.6) 208

Scanned with OKEN Scanner

4.1.4. Cazul ecuaţiilor cu diferenţe (2.12) 213
4.1.5. Cazul ecuaţiilor cu diferenţe cu corecţii pentru problema Dirichlet pentru ecuaţii eliptice slab neliniare 216
4.2. Metoda direcţiilor alternate 231
4.2.1. Algoritmul de calcul 231
4.2.2. Cazul ecuaţiilor cu diferenţe (2.4) 232
4.2.3. Cazul ecuaţiilor cu diferenţe (2.12) 232
4.2.4. Programul FORTRAN 234
4.2.5. Cazul ecuaţiilor cu diferenţe (2.4) 239
4.2.6. Cazul ecuaţiilor cu diferenţe (2.12) 241
4.3. Metoda lui Fridman 242
4.3.1. Algoritmul de calcul 243
4.3.2. Programul FORTRAN 243
4.4. Metoda gradientului 259
4.4.1. Algoritmul de calcul 259
4.4.2. Programul FORTRAN 260

Bibliografie 265

Scanned with OKEN Scanner

CAPITOLUL 1

REZOLVAREA NUMERICĂ A ECUAŢIILOR ALGEBRICE

Rezolvarea ecuaţiilor algebrice a frămîntat lumea matematică şi rămîne subiect de cercetare în continuare pentru matematicieni şi nu numai pentru matematicieni. In cele ce urmează prezentăm cîteva realizări recente în rezolvarea numerică a ecuaţiilor algebrice.

1.1. Metode de determinare a zerourilor polinoamelor algebrice

Metodele numerice clasice de determinare a zerourilor polinoamelor algebrice (metoda coardei, metoda secantei, metoda lui Newton, metoda lui Steffensen, metoda Cebîşev) sînt metode iterative. Sigur aceste metode au unele avantaje de necontestat : ordin de convergenţă mai mare ca 1, simplitatea programării metodelor ; dar au şi dezavantaje. Convergenţa locală a metodelor, de exemplu, implică determinarea unei iteraţii de start suficient de aproape de zeroul căutat, ceea ce poate conduce la un calcul laborios în cazul ecuaţiilor de grad mare, cînd avem de determinat toate zerourile.

Vom prezenta cîteva metode de "căutare" a zerourilor polinoamelor. Aceste metode au un ordin de convergenţă < 2 şi necesită un volum de calcul remarcabil. Sînt slab convergente, de aceea nu sînt indicate pentru a determina zerourile cu precizii mari, dar sînt eficiente pentru "localizarea" zerourilor cu pre-

cizii de ordinul $10^{-1}, 10^{-2}$. Pentru determinări mai precise se poate aplica una din metodele iterative menționate mai sus pentru zerourile care ne interesează, verificînd în prealabil dacă sînt îndeplinite condițiile de convergență locală a metodei iterative alese. Deoarece metodele iterative mai sus menționate sînt tratate în mai toate monografiile de analiză numerică [9(1), 17(1), 28(1), 40(2), 41(1), 41(2), 49(1), 65(1), 65(2)], aceste metode nu au fost prezentate.

1.2. Marginea zerourilor polinoamelor algebrice

Fie polinomul $P \in \mathbb{C}[z]$

$$P(z) = a_0 z^n + a_1 z^{n-1} + \ldots + a_n \quad , \quad a_0 \cdot a_n \neq 0 \qquad (1.1)$$

Notăm cu $z_1^*, z_2^*, \ldots, z_n^*$ zerourile polinomului P, adică $P(z_k^*) = 0$ pentru orice $k = \overline{1,n}$. Ne punem problema găsirii marginilor

$$m = \min \{|z_k^*| \ , \ k = \overline{1,n}\} \quad \text{și} \quad M = \max \{|z_k^*| \ , k=\overline{1,n}\} \qquad (1.2)$$

zerourilor polinomului (1.1). Pentru multiplele constante, ce se găsesc în bogata bibliografie la această temă [3(1), 52(1)], care mărginesc modulele zerourilor polinomului (1.1) prezentăm un algoritm cu care se pot îmbunătăți aceste constante oricît de mult dorim.

Teorema 1.1 (Cauchy). Dacă $P \in \mathbb{R}_+[x]$, $P(x) = a_0 x^n - a_1 x^{n-1} - \ldots - a_n$, cu $a_0 a_n \neq 0$, atunci P are un singur zerou pozitiv.

Demonstrație. $P(0) = -a_n < 0$, iar $\lim_{x \to \infty} P(x) = \infty \Rightarrow \exists \ x_0 \in \mathbb{R}_+$ astfel încît $P(x_0) = 0$. Fie $x > x_0$, adică $x = kx_0$ cu $k > 1$. Atunci

$$P(x) = P(kx_0) > k^n P(x_0) = 0 \ , \ \text{adică} \ \forall \ x > x_0 \ \ P(x) > 0.$$

Fie $x \in (0, x_0)$, adică $x = kx_0$ cu $0 < k < 1$. Atunci

$$P(x) = P(kx_0) < k^n P(x_0) = 0 \ , \ \text{adică} \ \forall \ x \in (0, x_0) \ P(x) < 0.$$

Rezultă că x_0 este unicul zerou pozitiv al polinomului P.

Observaţia 1.1. Fie $Q \in \mathbb{C}[z]$,

$$Q(z) = a_n z^n + a_{n-1} z^{n-1} + \ldots + a_0 \qquad (1.3)$$

unde a_i sînt coeficienţii polinomului (1.1), atunci $z_1^*, z_2^*, \ldots$ $\ldots, z_n^*$ sînt zerourile polinomului (1.1) dacă şi numai dacă $1/z_1^*, 1/z_2^*, \ldots, 1/z_n^*$ sînt zerourile polinomului Q .

Teorema 1.2 (Kakeya - 1912). Fie polinomul (1.1) şi R zeroul pozitiv al polinomului $Q_1 \in \mathbb{R}_+[x]$

$$Q_1(x) = |a_0| x^n - |a_1| x^{n-1} - \ldots - |a_n| \qquad (1.4)$$

Atunci

$$M \leq R$$

Demonstraţie. Dacă $Q_1(R) = 0$, conform teoremei 1.1 rezultă că pentru $\forall z \in \mathbb{C}$, $|z| > R$, $Q_1(|z|) > 0$, atunci avem că $|P(z)|$ $\geq Q_1(|z|) > 0$ pentru $\forall z \in \mathbb{C}$ cu $|z| > R$, adică $M \leq R$.

Corolarul 1. Fie polinomul (1.3) care are zerourile $1/z_1^*, 1/z_2^*, \ldots$ $\ldots, 1/z_n^*$ şi fie $1/r$ zeroul pozitiv al polinomului $Q_3(x) =$ $= |a_n| x^n - |a_{n-1}| x^{n-1} - \ldots - |a_0|$ atunci r este unicul zerou pozitiv al polinomului

$$Q_2(x) = |a_0| x^n + |a_1| x^{n-1} + \ldots + |a_{n-1}| x - |a_n| \qquad (1.5)$$

Conform teoremei 1.2 avem $1/|z_i^*| \leq 1/r$ pentru $\forall i = \overline{1,n}$ adică

$$r \leq m .$$

Teorema 1.3. Dacă notăm cu $b_k = |a_k| / |a_0|$ şi cu

$$M_1 = b_1 + \sqrt{b_2} + \ldots + \sqrt[n]{b_n}$$

$$M_2 = \max \{ 2b_1, 2\sqrt{b_2}, \ldots, 2\sqrt[n]{b_n} \}$$

$$M_3 = \max \{ nb_1, \sqrt{nb_2}, \ldots, \sqrt[n]{nb_n} \}$$

atunci pentru polinomul (1.4) avem : $Q_1(M_i) \geq 0$, $i = \overline{1,3}$.

Demonstrație.

a) $M_1^{n-1} = M_1^{k-1} M_1^{n-k} \geq (\sqrt[k]{b_k})^{k-1} \cdot M_1^{n-k}$; $\sqrt[k]{b_k} \cdot M_1^{n-1} \geq b_k M_1^{n-k}$, $\forall k = \overline{1,n}$

$Q_1(M_1)/|a_0| > M_1^n - b_1 \cdot M_1^{n-1} - \sqrt{b_2} \cdot M_1^{n-1} - \ldots - \sqrt[n]{b_n} \cdot M_1^{n-1} = M_1^n - M_1^{n-1} \cdot M_1 = 0$

b) $\sqrt[k]{b_k} \leq M_2$ atunci $b_k / M_2^k \leq 1/2^k$ $\forall k = \overline{1,n}$; $Q_1(M_2)/(|a_0| M_2^n) \geq$

$1 - b_1/M_2 - b_2/M_2^2 - \ldots - b_n/M_2^n \geq 1 - 1/2 - 1/2^2 - \ldots - 1/2^n = 1/2^n > 0$.

c) $\sqrt[k]{nb_k} \leq M_3$ atunci $b_k/M_3^k \leq \frac{1}{n}$; $Q_1(M_3)/(|a_0| M_3^n) \geq 1 - b_1/M_3 -$

$- b_2/M_3^2 - \ldots - b_n/M_3^n > 1 - 1/n - 1/n - \ldots - 1/n = 0$.

Corolarul 1. Deoarece $Q_1(R) = 0$ și $Q_1(M_i) \geq 0$ cu $i = \overline{1,3}$ atunci conform teoremei 1.1 rezultă că $R \leq M_i$ cu $i = \overline{1,3}$. Adică orice constantă $c \in R_+$ pentru care $Q_1(c) \geq 0$ va fi o margine superioară a zerourilor polinomului (1.1) și $R \leq c$.

Corolarul 2. Dacă considerăm polinomul (1.3) și notăm cu $C_k = |a_{n-k}|/|a_n|$, $k = \overline{1,n}$ și cu

$$m_1 = 1/(C_1 + \sqrt{C_2} + \ldots + \sqrt[n]{C_n})$$

$$m_2 = 1/\max\{2C_1, 2\sqrt{C_2}, \ldots, 2\sqrt[n]{C_n}\}$$

$$m_3 = 1/\max\{nC_1, \sqrt{nC_2}, \ldots, \sqrt[n]{nC_n}\}$$

atunci în mod analog rezultă : $m_i \leq r \leq m$.

Observația 1.2. Șirul lui Newton

$$\begin{cases} x_{k+1} = x_k - Q_1(x_k)/Q_1'(x_k) \\ x_0 = M_2 \end{cases} \qquad (1.6)$$

converge la R. Termenul x_k va aproxima prin adaos pe R. Condițiile de convergență sînt asigurate și anume :

a) $Q_1 \in \mathcal{C}^2(R)$

b) $Q_1'(x_0) \neq 0$

c) $J \subset R$, $J = [x_0 + 2h, x_0]$ unde $h = -Q_1(x_0)/Q_1'(x_0)$

d) $2bL \leq |Q_1'(x_0)|$ unde $L = \max\{Q_1''(x) | x \in J\}$ este îndeplinită pentru că este echivalentă cu inegalitatea :

$-2Q_1(x_o).Q_1'(x_o)/Q_1''(x_o) \leq |Q_1'(x_o)|$ care este evidentă.

Observaţia 1.3. Deoarece avem îndeplinite următoarele condiţii pentru polinomul (1.5) :

a) $Q_2''(x) > 0$ pentru $\forall x \in [o, M_2]$

b) $Q_2(M_2)Q_2''(M_2) > 0$

c) $Q_2(M_2).Q_2(0) < 0$,

sîntem în condiţiile de convergenţă ale metodei coardei, deci şirul :

$$\begin{cases} x_{k+1} = (x_o Q_2(x_k) - x_k Q_2(x_o))/(Q_2(x_k) - Q_2(x_o)) \\ x_o = M_2 \ , \quad x_1 = 0 \end{cases} \tag{1.7}$$

ne va furniza o aproximare prin lipsă a zeroului r al polinomului (1.5).

In continuare prezentăm fără demonstraţie o teoremă de existenţă a zerourilor polinomului (1.1) în discul unitate. Fie $P^* \in \mathbb{C}[z]$

$$P^*(z) = \bar{a}_o + \bar{a}_1 z + \ldots + \bar{a}_n z^n \tag{1.8}$$

unde cu $\bar{a}_i$ s-a notat conjugatul lui a_i . Fie operatorul $T : \mathbb{C}[z] \to \mathbb{C}[z]$

$$T(P(z)) = \bar{a}_n P(z) - a_o P^*(z).$$

Dacă notăm cu $b_i = \bar{a}_n a_i - a_o \bar{a}_{n-i}$, atunci

$$T(P(z)) = b_1 z^{n-1} + b_2 z^{n-2} + \ldots + b_n .$$

Pentru că $b_o = \bar{a}_n a_o - a_o \bar{a}_n = 0$ rezultă că gradul lui $T(P(z))$ va fi mai mic ca n . Aplicînd succesiv operatorul T după un număr $m < n+1$ de paşi, polinomul obţinut va fi polinomul nul. Se observă că $T(P(0)) = |a_n|^2 - |a_o|^2 \in \mathbb{R}$.

Teorema 1.4. Fie polinomul (1.1) şi m cel mai mic întreg pentru care $T^m(P(o)) = 0$.

a) Dacă există k , $1 \leq k < m$ astfel încît $T^k(P(o)) < 0$

atunci rezultă că polinomul (1.1) are cel puţin un zerou în discul $D(0,1) = \{ c \mid c \in \mathbb{C}, |c| < 1 \}$

b) Dacă $T^k(P(o)) > 0$ pentru $\forall k = \overline{1,m-1}$, iar $T^{m-1}(P(z))$ este o constantă reală, atunci rezultă că polinomul (1.1) nu are zerouri în $D(0,1)$, [37(1)].

Observaţia 1.4. Fie $c \in \mathbb{C}$, $r \in \mathbb{R}_+$, atunci $P(z)$ are zerouri în discul $D(c,r) = \{ z \mid z \in \mathbb{C}, |z - c| < r \}$ dacă şi numai dacă polinomul $P(rz + c)$ are zerouri în discul $D(0,1)$.

Observaţia 1.5. Fie polinomul (1.1) şi $Q(z) = P(rz + c)$. Dacă $T^m(Q(o)) = 0$ şi $T^k(Q(o)) > 0$ pentru $\forall k = \overline{1,m-1}$ şi $T^{m-1}(Q(z))$ constantă reală, teoreme nu precizează nimic.

In cele ce urmează după ce am stabilit marginea zerourilor polinoamelor din $\mathbb{C}[z]$, margini ce pot fi calculate analog şi pentru polinoame din $\mathbb{R}[x]$, pentru polinoame din $\mathbb{R}[x]$ ne punem problema determinării marginilor zerourilor reale. Fie $P \in \mathbb{R}[x]$:

$$P(x) = a_0x^n + a_1x^{n-1} + \ldots + a_n \quad \text{cu} \quad a_0 \cdot a_n \neq 0 \qquad (1.9)$$

Dacă polinomul (1.9) are zerouri multiple, după împărţirea cu c.m.m.d.c.(P,P') polinomul cît va avea zerouri distincte. De aceea putem considera în continuare că polinomul (1.9) are numai zerouri distincte.

Definiţia 1.1. Numim şir Sturm, următorul şir de polinoame

$$P_0, P_1, \ldots, P_s$$

care satisface următoarele condiţii :

a) P_i, P_{i+1} nu au zerouri comune pentru $\forall i = \overline{1,s-1}$

b) P_s nu are zerouri reale (păstrează semn constant)

c) Fie $x_0 \in \mathbb{R}$, astfel încît $P_1(x_0) = 0$, atunci $P_{i-1}(x_0) \cdot P_{i+1}(x_0) < 0$ pentru $\forall i = \overline{1,s-1}$

d) Dacă notăm cu $Q(x) = P_0(x)P_1(x)$ atunci pentru $x_0 \in \mathbb{R}$ astfel încît $P(x_0) = 0$, va exista $\varepsilon > 0$ astfel încît :

$Q(x) < 0$ pentru $\forall x \in (x_0-\varepsilon, x_0)$ şi $Q(x) > 0$ pentru $\forall x \in (x_0, x_0+\varepsilon)$, sau $Q(x) > 0$ pentru $\forall x \in (x_0-\varepsilon, x_0)$ şi $Q(x) < 0$ pentru $\forall x \in (x_0, x_0+\varepsilon)$.

Lema 1.1. Pentru orice polinom $P \in \mathbb{R}[x]$ există şirul Sturm.

Demonstraţie. Pentru demonstrarea lemei se construieşte următorul şir de polinoame : $P_0(x) = P(x)$, $P_1(x) = P'(x), \ldots, P_i(x) = -R_i(x)$ unde $R_i(x) = P_{i+2}(x) - P_{i+1}(x) \cdot Q_{i+1}(x), \ldots, P_s(x) = -R_s(x)$ unde $R_s(x) = P_{s+2}(x) - P_{s+1}(x) Q_{s+1}(x)$. Prin Q_{i+1} este notat cîtul împărţirii lui P_{i+2} la P_{i+1}. Acest şir de polinoame este cel pe care îl calculăm în algoritmul lui Euclid pentru aflarea c.m.m.d.c (P, P') cu observaţia că resturile se vor înmulţi cu -1. Se verifică uşor că este un şir Sturm. De remarcat că acest şir nu este unicul şir Sturm ataşat polinomului P. Pentru un polinom P şi şirul Sturm asociat lui P definim funcţia $S : \mathbb{R} \to \mathbb{N}$ care asociază $\forall x \in \mathbb{R}$ numărul de variaţii de semn din şirul de numere $P_0(x), P_1(x), \ldots, P_s(x)$.

Amintim (fără a o demonstra) teorema de existenţă a zerourilor reale într-un interval $(a,b) \subset \mathbb{R}$ [52(1)] :

Teorema 1.5 (Sturm) Pentru $\forall a, b \in \mathbb{R}$, $a < b$ astfel încît $P(a) \neq 0$ şi $P(b) \neq 0$ avem :

a) $S(a) \geq S(b)$

b) Numărul de zerouri reale din intervalul (a,b) este egal cu $S(a) - S(b)$.

Algoritmul 1.1. Cu ajutorul teoremei (1.4) se poate defini un algoritm care va furniza un şir de numere ce aproximează prin lipsă pe m (1.2) cu precizia ε, excepţie făcînd cazul precizat la observaţia 1.5. Fie polinomul (1.1).

Pasul 1. Fie $k=0$, $r_k = r$, unde r a fost determinat cu formulele (1.7) şi $q = 1$.

Pasul 2. - Dacă $P((r_k + q)z)$ nu are zerouri în discul $D(0,r_k+q)$ atunci $r_{k+1} = r_k + q$ și repetă pasul 2.

- Dacă $P((r_k + q)z)$ are zerouri în discul $D(0,r_k+ q)$ atunci :

  - Dacă $q < \varepsilon$ atunci r_k va fi o aproximație prin lipsă a lui m (1.2) de precizie ε ; se oprește algoritmul.

  - Dacă $q \geqslant \varepsilon$ atunci $q = q/10$ și repetă pasul 2.

- Dacă despre zerourile polinomului $P((r_k + q)z)$ nu se poate preciza nimic în discul $D(0,r_k+ q)$ atunci :

  - Dacă $q < \varepsilon$ atunci r_k va fi o aproximație prin lipsă (nu neapărat de precizie ε) a lui m (1.2) ; se oprește algoritmul

  - Dacă $q \geqslant \varepsilon$ facem $q = q/10$ și repetă pasul 2.

Observația 1.6. Pentru îmbunătățirea lui R (1.6) vom considera polinomul (1.3) și $r_0 = 1/R$ în pasul 1, iar la sfîrșitul algoritmului (1.1) $1/r_k$ va fi o aproximație prin adaos a lui M (1.2).

Algoritmul 1.2. Notăm cu Z_r mulțimea zerourilor reale ale polinomului (1.9) și fie $Z_{r-} = Z_r \cap \mathbb{R}_-$ și $Z_{r+} = Z_r \cap \mathbb{R}_+$ mulțimea zerourilor reale negative respectiv pozitive. Evident $Z_{r-} \subset [-M, -m]$ și $Z_{r+} \subset [m,M]$. Cu ajutorul teoremei (1.5) (Sturm) se poate defini un algoritm care va furniza intervalele ale căror capete diferă cu o mărime mai mică decît ε (precizia impusă) față de $\min Z_{r-}$, $\max Z_{r-}$, $\min Z_{r+}$, $\max Z_{r+}$.

Fie (a,b) intervalul în discuție cu $P(a) \neq 0$ și $P(b) \neq 0$.

Pasul 1. Fie $q = 1$ și $N = S(a) - S(b)$ atunci :

- Dacă $N = 0$, $a = b$ și algoritmul se oprește. În intervalul inițial (a,b) nu avem zerouri reale.

- Dacă $N > 0$ se continuă cu pasul 2 ;

Scanned with OKEN Scanner

Pasul 2. - Dacă $P(a + q) = 0$ atunci $q = q - \varepsilon$ şi :

- Dacă $q < \varepsilon$, luăm $q = 1$ şi se continuă cu pasul 3

- Dacă $q \geq \varepsilon$ se reia pasul 2

- Dacă $P(a + q) \neq 0$ atunci :

- Dacă $S(a + q) - S(b) = N$, se pune $a = a + q$ şi se repetă pasul 2.

- Dacă $S(a + q) - S(b) < N$, atunci luăm $q = q/10$ şi

- Dacă $q < \varepsilon$, se ia $q = 1$ şi se continuă cu pasul 3

- Dacă $q \geq \varepsilon$ repetăm pasul 2.

Pasul 3. - Dacă $P(b - q) = 0$ atunci $q = q - \varepsilon$ şi

- Dacă $q < \varepsilon$ se opreşte algoritmul, intervalul (a,b) va fi cel căutat.

- Dacă $q \geq \varepsilon$ repetăm pasul 3.

- Dacă $P(b - q) \neq 0$ atunci :

- Dacă $S(a) - S(b-q) = N$, luăm $b = b - q$ şi repetăm pasul 3

- Dacă $S(a) - S(b-q) < N$, luăm $q = q/10$ şi

- Dacă $q < \varepsilon$ se opreşte algoritmul, intervalul (a,b) va fi cel căutat.

- Dacă $q \geq \varepsilon$ repetăm pasul 3.

Textul sursă al programului KAKEYA reuneşte algoritmul (1.1) şi algoritmul (1.2). Programul calculează şi constantele R, r, M_1, m_1, M_2, m_2, M_3, m_3 şi eventual M_r şi m_r (marginile rădăcinilor reale ale polinoamelor din $\mathbb{R}[X]$).

```
      PROGRAM KAKEYA
      COMPLEX A,Z
      REAL M,M1,M2,M3,MM,MM1,MM2,MM3
      DIMENSION A(Ø:15),Q(Ø:15)
      COMMON/E1/A,T,E,KD/E2/N/E4/M,MM
      CALL ASSIGN(1,'SY:KAKEYA.DAT')
      CALL ASSIGN(2,'SY:KAKEYA.LST')
      READ(1,*) N,E
      READ(1,1,END=2) (A(I),I=Ø,N)
1     FORMAT(6F)
      WRITE(2,21)  (I,A(I),I=Ø,N)
21    FORMAT(1X,'CALCULUL MARGINILOR ZEROURILOR POLINOMULUI'//
     1<N+1>(' a(',I2,')=(',F,',',F,')'/))
2     DO 3 I=Ø,N
3     Q(I)=CABS(A(I))
      M1=Q(1)/Q(Ø)
      M2=2.*M1
      S=FLOAT(N)
      M3=S*M1
      DO 4 I=2,N
      X=(Q(I)/Q(Ø))**(1./I)
      Y=X*S**(1./I)
      M1=M1+X
      X2=2.*X
      IF(M2.LT.X2) M2=X2
      IF(M3.LT.Y) M3=Y
4     CONTINUE
      MM1=Q(N-1)/Q(N)
      MM2=2.*MM1
      MM3=S*MM1
```

Scanned with OKEN Scanner

```
      DO 5 I=2,N
      X=(Q(N-I)/Q(N))**(1./I)
      Y=X**S**(1./I)
      MM1=MM1+X
      X2=2.*X
      IF(MM2.LT.X2) MM2=X2
      IF(MM3.LT.Y) MM3=Y
5     CONTINUE
      MM1=1./MM1
      MM2=1./MM2
      MM3=1./MM3
      R=M2
6     Q1=R*Q(Ø)
      DO 7 I=1,N-1
7     Q1=(Q1-Q(I))*R
      Q1=Q1-Q(N)
      Q11=R*S*Q(Ø)
      DO 8 I=1,N-2
8     Q11=(Q11-(N-I)*Q(I))*R
      Q11=Q11-Q(N-1)
      QQ=Q1/Q11
      IF(QQ.LT.E)GOTO 9
      R=R-QQ
      GOTO 6
9     R=R-QQ
      RM=R
1Ø    Q2Ø=RM*Q(Ø)
      DO 11 I=1,N-1
11    Q2Ø=(Q2Ø+Q(I))*RM
      Q2Ø=Q2Ø-Q(N)
```

Scanned with OKEN Scanner

```
      IF(Q2Ø.LT.Ø.) GOTO 12
      Q22=Q2Ø
      RM=RM/2.
      GOTO 1Ø
12    Q2Ø=Q22
      X1=RM*Q(N)/(Q(N)+Q2Ø)
13    Q2X=X1*Q(Ø)
      DO 14 I=1,N-1
14    Q2X=(Q2X+Q(I))*X1
      Q2X=Q2X-Q(N)
      XØ=(RM*Q2X-X1*Q2Ø)/(Q2X-Q2Ø)
      IF(XØ-X1.LT.E) GOTO 15
      X1=XØ
      GOTO 13
15    RM=XØ
      T=RM
      CALL ALG
      MM=T
      DO 2Ø I=Ø,N/2
      Z=A(I)
      A(I)=A(N-I)
2Ø    A(N-I)=Z
      T=1./R
      CALL ALG
      M=1./T
      DO 24 I=Ø,N/2
      Z=A(I)
      A(I)=A(N-I)
24    A(N-I)=Z
      WRITE(2,16) M1,MM1,M2,MM2,M3,MM3,R,RM,E
```

Scanned with OKEN Scanner

```
16    FORMAT(1X,'M1 = ',F,3X,'m1 = ',F/
      31X,'M2 = ',F,3X,'m2 = ',F/
      41X,'M3 = ',F,3X,'m3 = ',F/
      51X,' R = ',F,3X,' r = ',F,3X,'cu precizia=',F7.6)
      IF(KD.EQ.1) GOTO 17
      WRITE(2,18) M,MM,E
18    FORMAT(1X,'M=',F,3X,'m=',F,3X,'cu precizia=',F7.6)
      KI=0
      DO 23 I=0.N
      IF(AIMAG(A(I)).NE.0.) KI=1
23    CONTINUE
      IF(KI.EQ.1) STOP
      WRITE(2,25)
25    FORMAT(//1X,'CALCULUL MARGINILOR ZEROURILOR REALE'//)
      CALL STKA
      STOP
17    WRITE(2,19) M,MM
19    FORMAT(1X,' M = ',F,3X,' m = ',F)
      KI=0
      DO 22 I=0,N
      IF(AIMAG(A(I)).NE.0.) KI=1
22    CONTINUE
      IF(KI.EQ.1) STOP
      WRITE(2,25)
      CALL STKA
      STOP
      END
      SUBROUTINE ALG
      COMPLEX A,B
      DIMENSION A(0:15),B(0:15)
```

Scanned with OKEN Scanner

```
      COMMON/E1/A,T,E,KD/E2/N/E3/B,KOD
      D=Ø.1
1     XIM=1.7E38
      XAM=Ø.
      KD=Ø
      DO 2 I=Ø,N
      B(I)=A(I)*(T+D)**(N-I)
      X=CABS(B(I))
      IF(X.LT.XIM.AND.X.NE.Ø.) XIM=X
      IF(X.GT.XAM) XAM=X
2     CONTINUE
      Y=SQRT(XIM)*SQRT(XAM)
      DO 3 I=Ø,N
3     B(I)=B(I)/Y
      CALL SCHUR
      IF(KOD.EQ.2) KD=1
      GOTO(4,5,4),KOD+1
4     IF(D.LT.E) GOTO 6
      D=D/1Ø.
      GOTO 1
5     T=T+D
      GOTO 1
6     RETURN
      END
      SUBROUTINE SCHUR
      COMPLEX A,B
      DIMENSION A(Ø:15),B(Ø:15)
      COMMON/E2/N/E3/B,KOD
      IF(B(N).EQ.(Ø.,Ø.)) GOTO 1Ø
      M=1
      K=N
```

Scanned with OKEN Scanner

```
      L=1
      KM=Ø
8     MT1=L
      L=Ø
      DO 2 I=Ø,K-1
      A(I)=CONJG(B(K))*B(I)-B(Ø)*CONJG(B(K-I))
      IF(A(I).NE.(Ø.,Ø.)) L=1
2     CONTINUE
      A(K)=CONJG(B(K))*B(K)-B(Ø)*CONJG(B(Ø))
      XIM=1.7E38
      XAM=Ø.
      DO 9 I=1,K
      X=CABS(A(I))
      IF(X.LT.XIM.AND.X.NE.Ø.) XIM=X
      IF(X.GT.XAM) XAM=X
9     CONTINUE
      Y=SQRT(XIM)*SQRT(XAM)
      IF(REAL(A(K)).EQ.Ø.) GOTO 6
      IF(REAL(A(K)).LT.Ø.) KM=M
      B(K)=(Ø.,Ø.)
      DO 7 I=Ø,K-1
      A(I+1)=A(I+1)/Y
7     B(I)=A(I+1)
      M=M+1
      K=K-1
      IF(K.GE.1) GOTO 8
      MT1=Ø
6     IF(KM.NE.Ø) GOTO 11
      IF(MT1.EQ.Ø) GOTO 12
      KOD=2
      RETURN
```

Scanned with OKEN Scanner

```
11      KOD=Ø
        RETURN
12      KOD=1
        RETURN
1Ø      WRITE(2,21)
21      FORMAT(1X,'P(Ø)=Ø EROARE')
        KOD=3
        RETURN
        END
        SUBROUTINE STKA
        COMPLEX A
        REAL M,MM
        DIMENSION A(Ø:15),U(Ø:15)
        COMMON/E1/A,T,E,KD/E2/N/E4/M,MM/E5/U,NN/E7/A1,B1,NØ
        DO 1 I=Ø,N
1       U(I)=REAL(A(I))
        NN=N
        CALL WORMA(U,N)
        CALL SSTURM
        NN=N
        A1=-M
        B1=-MM
        DO 2 I=Ø,N
2       U(I)=REAL(A(I))
        CALL WORMA(U,N)
        CALL ALG1
        IF(NØ.EQ.Ø) GOTO 3
        IF(KD.EQ.1) GOTO 14
        WRITE(2,4) A1,B1,E
4       FORMAT(1X,'MR=',F,3X,'mr=',F,3X,3X,'cu precizia=',F7.6)
        GOTO 3
```

Scanned with OKEN Scanner

```
14    WRITE(2,5)A1,B1
5     FORMAT(1X,'MR =',F,3X,'mr = ',F)
3     N=NN
      A1=MM
      B1=M
      DO 6 I=Ø,N
6     U(I)=REAL(A(I))
      CALL WORMA(U,N)
      CALL ALG1
      IF(NØ.EQ.Ø) GOTO 7
      IF(KD.EQ.1) GOTO 8
      WRITE(2,9) A1,B1,E
9     FORMAT(1X,'mr = ',F,3X,'MR = ',F,3X,'cu precizie = ',F7.6)
      GOTO 7
8     WRITE(2,1Ø) A1,B1
1Ø    FORMAT(1X,'mr = ',F,3X,'MR = ',F)
7     RETURN
      END
      SUBROUTINE SSTURM
      DIMENSION U(Ø:15),V(Ø:15),Q(Ø:15),
      1S(Ø:15,Ø:15),N1(Ø:15)
      COMMON/E5/U,N/E6/S,N1
      DO 1 I=Ø,N-1
      S(Ø,I)=U(I)
1     V(I)=(N-I)*U(I)
      S(Ø,N)=U(N)
      CALL WORMA(V,N-1)
      DO 1Ø I=Ø,N-1
1Ø    S(1,I)=V(I)
      N1(Ø)=N
      N1(1)=N-1
```

Scanned with OKEN Scanner

```
      DO 11 I=2,15
      DO 11 K=Ø,15
11    S(I,K)=Ø.
      DO 12 I=2,15
12    N1(I)=-1
      M=N-1
      L=2
2     DO 3 K=Ø,N-M
      Q(K)=U(K)/V(Ø)
      DO 3 J=K+1,M+K
3     U(J)=U(J)-Q(K)*V(J-K)
      J=N-M+1
      DO 4 I=Ø,M-1
4     Q(I)=U(J+I)
      I=-1
5     I=I+1
      IF(ABS(Q(I)).LE..1E-6.AND.I.LT.M-1) GOTO 5
      K=M-1-I
      N1(L)=K
      DO 7 K=Ø,M
7     U(K)=V(K)
      N=M
      DO 8 K=I,M-1
      V(K-I)=-Q(K)
8     S(L,K-I)=-Q(K)
      M=N1(L)
      IF(L.GE.N1(Ø).OR.M.EQ.Ø) GOTO 9
      L=L+1
      GOTO 2
9     RETURN
      END
```

Scanned with OKEN Scanner

```
      SUBROUTINE ALG1
      COMPLEX AA
      DIMENSION U(Ø:15),AA(Ø:15)
      COMMON/E1/AA,T,E,KD/E5/U,N/E7/A,B,NØ
      Q=1.
      NA=KAV(A)
      NB=KAV(B)
      NØ=NA-NB
      IF(NØ.GT.Ø) GOTO 1
      A=D
      RETURN
1     X=U(Ø)
      KD=Ø
      DO 2 I=1,N
2     X=X*(A+Q)+U(I)
      IF(ABS(X).LE..1E-6) GOTO 4
      NAA=KAV(A+Q)
      IF(NAA-NB.LT.NØ) GOTO 3
      A=A+Q
      GOTO 1
3     Q=Q/1Ø.
      IF(Q.GE.E) GOTO 1
      Q=1.
      GOTO 5
4     Q=Q-E
      KD=1
      IF(Q.GE.E) GOTO 1
      Q=1.
5     X=U(Ø)
      DO 6 I=1,N
6     X=X*(B-Q)+U(I)
```

Scanned with OKEN Scanner

```
      IF(ABS(X).LE..1E-6) GOTO 8
      NBB=KAV(B-Q)
      IF(NA-NBB.LT.NØ) GOTO 7
      B=B-Q
      GOTO 5
7     Q=Q/1Ø.
      IF(Q.GE.E) GOTO 5
      RETURN
8     Q=Q-E
      KD=1
      IF(Q.GE.E) GOTO 5
      RETURN
      END
      FUNCTION KAV(X)
      DIMENSION S(Ø:15,Ø:15),N1(Ø:15)
      COMMON/E6/S,N1
      KAV=Ø.
      A=S(Ø,Ø)
      DO 1 I=1,N1(Ø)
1     A=A*X+S(Ø,I)
      J=1
3     B=S(J,Ø)
      DO 2 I=1,N1(J)
2     B=B*X+S(J,I)
      IF(A*B.LT.Ø.) KAV=KAV+1
      A=B
      J=J+1
      IF(N1(J).GE.Ø) GOTO 3
      RETURN
      END
```

Scanned with OKEN Scanner

```
      SUBROUTINE WORMA(A,N)
      DIMENSION A(Ø:15)
      XIM=Ø.17E+39
      XAM=Ø.
      DO 1 I=Ø,N
      X=ABS(A(I))
      IF(X.LT.XIM.AND.X.NE.ϕ.) XIM=X
      IF(X.GT.XAM) XAM=X
1     CONTINUE
      X=SQRT(XIM*XAM)
      DO 2 I=Ø,N
2     A(I)=A(I)/X
      RETURN
      END
```

Exemple rulate :

I. CALCULUL MARGINILOR ZEROURILOR POLINOMULUI

a(Ø) = (1.,Ø.)

a(1) = (4.,-12.)

a(2) = (-46.,-48.)

a(3) = (-124.,2ØØ.)

a(4) = (129.,8Ø.)

a(5) = (-624.,5812.)

a(6) = (11796.,158Ø8.)

a(7) = (29184.,-2184Ø.)

a(8) = (-4Ø32Ø.,Ø.)

M1 = 49.6ØØ2ØØ7 m1 = Ø.2732842

M2 = 25.2982216 m2 = Ø.1366421

M3 =1Ø1.1928864 m3 = Ø.138267Ø

R = 17.3651848 r = Ø.5424504 e = Ø.ØØØ1

M = 8.ØØØ864Ø r = Ø.9999505 e = Ø.ØØØ1

Scanned with OKEN Scanner

II. CALCULUL MARGINILOR ZEROURILOR POLINOMULUI (Gauss)

a(0) = (3432.,0.)

a(1) = (-12012.,0.)

a(2) = (16632.,0.)

a(3) = (-11550.,0.)

a(4) = (4200.,0.)

a(5) = (-756.,0.)

a(6) = (56.,0.)

a(7) = (1.,0.)

M1 = 9.8068457 m1 = 0.0080021

M2 = 7.0000000 m2 = 0.0040011

M3 = 24.5000000 m3 = 0.0025510

R = 4.6966658 r = 0.0091211 e = 0.0001

M = 0.9646762 m = 0.0146211 e = 0.0001

MR = -0.0156761 mr = -0.0146211 e = 0.0001

III. CALCULUL MARGINILOR ZEROURILOR POLINOMULUI

a(0) = (1.4121000 , 1.7320499)

a(1) = (-2.2360599 , 3.1622701)

a(2) = (1.2599200 , 1.4422400)

a(3) = (-1.1892000 , -0.6931400)

a(4) = (1.0986100 , 2.3025801)

a(5) = (0.4342900 , -3.1415901)

a(6) = (-0.3183000 , 9.8696003)

M1 = 6.8949432 m1 = 0.2761232

M2 = 3.4640968 m2 = 0.1380616

M3 = 10.3922901 m3 = 0.5189362

R = 2.1846920 r = 0.5959163 e = 0.0001

M = 1.9664090 m = 1.0673164 e = 0.0001

Scanned with OKEN Scanner

1.3. Determinarea simultană a zerourilor reale ale polinoamelor din $\mathbb{R}[x]$

Fie polinomul P(1.9) şi notăm cu $Z = \{x_1^*, x_2^*, \ldots, x_n^*\}$ mulţimea zerourilor polinomului P, adică $P(x_k^*) = 0$ pentru orice $k = \overline{1,n}$. Următorul algoritm, care are la bază teorema (1.5)(Sturm), determină simultan toate zerourile reale ale polinomului P(1.9), metoda fiind de generalitate maximă. Pentru simplitatea expunerii notăm cu P_n polinomul P de gradul n.

Algoritmul 1.3.

Pasul 1. - Dacă $n \leq 2$ se rezolvă ecuaţia $P_n(x) = 0$ cu ajutorul formulelor şi se opreşte algoritmul.

- Dacă $n > 2$ se continuă cu pasul 2.

Pasul 2. Calculăm $Q_{n-m}(x)$ = c.m.m.d.c. (P_n, P_n').

- Dacă $n-m > 0$ atunci $P_m = P_n/Q_{n-m}$ şi continuăm cu pasul 3.

- Dacă $n-m = 0$, $m = n$ şi trecem la pasul 3.

Pasul 3. Determinăm şirul Sturm asociat polinomului P_m, apoi, pasul 4.

Pasul 4. Determinăm cu ajutorul algoritmului (1.2) intervalele de lungime minimă (D_1,d_1), (d_2,D_2) şi numerele N_1, N_2 de zerouri reale negative respectiv pozitive ale polinomului P_m. (Dacă $N_i = 0$, convenim ca $D_i = d_i$). Urmează pasul 5.

Pasul 5. Fie stiva J care conţine la fiecare poziţie următoarele informaţii : capetele a şi b ale intervalelor care conţin zerouri reale ale polinomului P_m şi N numărul de zerouri reale din interval. Iniţial stiva J va conţine intervalele (D_1,d_1), (d_2,D_2) şi numerele N_1 şi N_2 numai dacă $N_i \neq 0$ pentru $i = 1$ sau $i = 2$.

- Dacă $P_m((a+b)/2) = 0$ atunci $P_{m-1}(x) = P_m(x)/(x-(a+b)/2)$, $m = m-1$ şi :

- Dacă $m \leq 2$ rezolvăm ecuația $P_m(x) = 0$ cu ajutorul formulelor și se continuă cu pasul 8
- Dacă $m > 2$ reluăm algoritmul cu pasul 3

- Dacă $P_m((a + b)/2) \neq 0$ atunci
  - Dacă $S(a)-S((a + b)/2) = 0$ se înlocuiește intervalul (a,b) cu intervalul $((a + b)/2,b)$ și $N = S((a + b)/2) - S(b)$. Urmează pasul 6.
  - Dacă $S(a)-S((a + b)/2) > 0$ și :
    - Dacă $S((a + b)/2)-S(b) > 0$ atunci se adaugă în stiva J intervalul $((a + b)/2,b)$ și $N = S((a+b)/2)-S(b)$ și se înlocuiește intervalul (a,b) cu intervalul $(a,(a + b)/2$ iar $N = S(a)-S((a + b)/2$. Urmează pasul 6.
    - Dacă $S((a + b)/2)-S(b) = 0$ înlocuim intervalul (a,b) cu intervalul $(a,(a + b)/2)$, $N = S(a) - S((a + b)/2)$ și continuăm cu pasul 6.

Pasul 6. Repetăm pasul 5 cu intervalele de lungime $> \varepsilon$ atîta timp cît există astfel de intervale în stiva J. La terminarea repetiției se execută pasul 7.

Observația 1.7. Numărul pozițiilor din stiva J nu este mai mare decît gradul polinomului P.

Pasul 7. Putem considera că zerourile reale ale polinomului P_m, determinate cu precizia ε, sînt cele p ($p \leq m$) numere reale x, obținute din cele ℓ ($\ell \leq p$) intervale (a_k,b_k) după cum urmează : fie intervalul (a_k,b_k) cel în discuție și N_k numărul zerourilor reale din acest interval. Considerăm că : $h_k = (b_k - a_k)/(N_k + 1)$, atunci

$$x_{k,j} = a_k + j.h_k \qquad j = \overline{1,N_k} \quad \text{și} \quad k = \overline{1,\ell}$$

Pasul 8. - Dacă gradul $n-m > 0$, atunci se reia algoritmul cu polinomul Q_{n-m} de la pasul 1.

- Dacă gradul n-m = o algoritmul se opreşte. S-au obţinut toate zerourile reale ale polinomului P_m cu precizia ε.

Convergenţa metodei este asigurată, deoarece un număr finit de intervale de lungime finită sînt înjumătăţite la fiecare repetiţie a pasului 5. Evident, după un număr finit de paşi, lungimile intervalelor vor fi mai mici decît precizia impusă. Programul STURM descrie acest algoritm cu o eficienţă bună în găsirea zerourilor.

Ordinul de convergenţă al metodei

Fie $\{x_i^k\}$ $i = \overline{1,n}$ cele n şiruri de numere ce reprezintă mijloacele intervalelor în care se află zerourile x_i^* $i = \overline{1,n}$. Atunci avem

$$x_i^k \longrightarrow x_i^* \quad \text{pentru} \quad i = \overline{1,n} \quad \text{cînd} \quad k \longrightarrow \infty$$

Atîta timp cît $|x_i^{k+1} - x_i^k| \geq \varepsilon$ rezultă că $|x_i^{k+1} - x_i^*| \geq \varepsilon$ pentru $\varepsilon > 0$. Pentru zerourile x_i^* negative avem inegalitatea $|x_i^k - x_i^*| \leq \frac{D_1 - d_1}{2}$, iar pentru zerourile x_i^* pozitive avem inegalitatea $|x_i^k - x_i^*| \leq \frac{D_2 - d_2}{2}$. Deci

$$\frac{|x_i^k - x_i^*|}{|x_i^{k+1} - x_i^*|} \leq \frac{D_1 - d_1}{2\varepsilon} \quad \text{sau} \quad \frac{|x_i^k - x_i^*|}{|x_i^{k+1} - x_i^*|} \leq \frac{D_2 - d_2}{2\varepsilon},$$

după cum x_i^* este negativ sau pozitiv. Inegalităţile de mai sus ne permit să afirmăm că ordinul de convergenţă al algoritmului STURM este cel puţin 1.

```
      PROGRAM STURM
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      CHARACTER DA*2
      DIMENSION A(0:30),B(0:30),U(0:30),E(30),F(30),NR(30)
      COMMON/B1/U,N/B3/A1,B1,N0/B4/EP/B5/A,NA
      CALL ASSIGN(1,'TI:')
```

```
      CALL ASSIGN(2,'SY:STURM.DAT')
      CALL ASSIGN(3,'SY:STURM.LST')
      READ(2,1) NN,EP,DA
1     FORMAT(I5,F16.15,A2)
      N=NN
      READ(2,2,END=3)(A(I),I=Ø,N)
2     FORMAT(4F)
      IF(DA.EQ.'DA') NA=1
      WRITE(1,4) EP,(I,A(I),I=Ø,N)
4     FORMAT(' ','DETERMINAREA RADACINILOR REALE A ECUATIEI P(X)=Ø'/
      15X,'PRECIZIA IMPUSA:',F/
      25X,'COEFICIENTII POLINOMULUI'/
      3<N+1>(1X,'A(',I2,') = ',F/),
      41X,'RADACINILE ECUATIEI')
      IF(NA.EQ.1) WRITE(3,4) EP,(I,A(I),I=Ø,N)
5     IF(N.EQ.Ø) STOP
      CALL WORMA(A,N)
      GOTO(6,7),N
      DO 9 I=Ø,N
9     U(I)=A(I)
      CALL EUCLID
      IF(N.EQ.Ø) GOTO 12
      DO 1Ø K=Ø,NN-N
      B(K)=A(K)/U(Ø)
      DO 1Ø J=K+1,N+K
1Ø    A(J)=A(J)-B(K)*U(J-K)
      DO 11 I=Ø,N
11    B(I)=U(I)
      NN=NN-N
      CALL WORMA(A,NN)
12    M=N
```

Scanned with OKEN Scanner

```
      N=NN
      GOTO(26,27),N
      GOTO 8
26    CALL EC1
      DO 28 I=Ø,M
28    A(I)=B(I)
      NN=M
      N=M
      GOTO 5
27    CALL EC2
      DO 29 I=Ø,M
29    A(I)=B(I)
      NN=M
      N=M
      GOTO 5
8     DO 13 I=Ø,N
13    U(I)=A(I)
      CALL SSTURM
      N=NN
      AØ=DABS(A(Ø))
      AN=DABS(A(N))
      T=2.*DABS(A(1))/AØ
      S=2.*DABS(A(N-1))/AN
      DO 14 I=2,N
      X=2.*(DABS(A(I))/AØ)**(1./I)
      Y=2.*(DABS(A(N-I))/AN)**(1./I)
      IF(T.LT.X) T=X
      IF(S.LT.Y) S=Y
14    CONTINUE
      S=1./S
      DO 141 K=Ø,3Ø
```

Scanned with OKEN Scanner

```
      E(K)=.ØDØ
141   F(K)=.ØDØ
      DO 3Ø I=Ø,N
3Ø    U(I)=A(I)
      A1=-T
      B1=-S
      K=Ø
      CALL ALG1
      IF(NØ.EQ.Ø) GOTO 142
      K=1
      E(K)=A1
      F(K)=B1
      NR(K)=NØ
142   A1=S
      B1=T
      CALL ALG1
      IF(NØ.EQ.Ø) GOTO 143
      K=K+1
      E(K)=A1
      F(K)=B1
      NR(K)=NØ
143   MAR=1
      WRITE(3,7Ø1) (E(J),F(J),J=1,K)
7Ø1   FORMAT(1X,'RADACINILE REALE ALE POLINOMULUI SE AFLA IN'/
     11X,'INTERVALELE:',('(',F,',',F,')'))
15    IF(MAR.EQ.Ø) GOTO 23
      MAR=Ø
      J=1
16    IF(J.GT.K) GOTO 15
      IF(F(J)-E(J).GE.EP.AND.NR(J).NE.Ø) GOTO 17
      J=J+1
```

Scanned with OKEN Scanner

```
      GOTO 16
17    X=(E(J)+F(J))/2.
      MAR=1
      P=A(Ø)
      DO 18 I=1,N
18    P=P*X+A(I)
      IF(P.EQ..ØDØ) GOTO 21
      NE=KAV(E(J))
      NEF=KAV(X)
      IF(NE-NEF.GT.Ø) GOTO 19
      E(J)=X
      J=J+1
      GOTO 16
19    NF=KAV(F(J))
      IF(NEF-NF.EQ.Ø) GOTO 2Ø
      K=K+1
      E(K)=X
      F(K)=F(J)
      NR(K)=NEF-NF
      F(J)=X
      NR(J)=NE-NEF
      J=J+1
      GOTO 16
2Ø    F(J)=X
      J=J+1
      GOTO 16
21    DO 22 I=1,N
22    A(I)=A(I)+A(I-1)*X
      N=N-1
      WRITE(1,221) X
221   FORMAT(' ','RADACINA REALA EXACTA:',F)
```

Scanned with OKEN Scanner

```
      IF(NA.EQ.1) WRITE(3,221) X
      CALL WORMA(A,N)
      NN=N
      IF(N.GT.2) GOTO 8
      CALL EC2
      DO 222 I=Ø,M
222   A(I)=B(I)
      N=M
      NN=M
      GOTO 5
23    DO 24 J=1,K
      H=(F(J)-E(J))/(NR(J)+1)
      DO 24 L=1,NR(J)
      X=E(J)+L*H
      WRITE(1,231) X
231   FORMAT(' ','RADACINA REALA APROXIMATIVA:',F)
      IF(NA.EQ.1) WRITE(3,231) X
24    CONTINUE
      N=M
      DO 25 I=Ø,N
25    A(I)=B(I)
      NN=M
      GOTO 5
6     CALL EC1
      STOP
7     CALL EC2
      STOP
      END
      SUBROUTINE EC1
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
```

```
      DIMENSION A(Ø:3Ø)
      COMMON/E5/A,NA
      X=-A(1)/A(Ø)
      WRITE(1,1) X
1     FORMAT(' ','RADACINA REALA EXACTA:',F)
      IF(NA.EQ.1) WRITE(3,1)X
      RETURN
      END
      SUBROUTINE EC2
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION A(Ø:3Ø)
      COMMON/E5/A,NA
      D=A(1)*A(1)-4.*A(Ø)*A(2)
      IF(DABS(D).LT.1D-13) D=.ØDØ
      IF(D.LT..ØDØ) GOTO 2
      D=DSQRT(D)
      X=(-A(1)-D)/(2.*A(Ø))
      WRITE(1,1) X
1     FORMAT(' ','RADACINA REALA EXACTA:',F)
      IF(NA.EQ.1) WRITE(3,1) X
      X=(-A(1)+D)/(2.*A(Ø))
      WRITE(1,1) X
      IF(NA.EQ.1) WRITE(3,1) X
2     RETURN
      END
      SUBROUTINE WORMA(A,N)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION A(Ø:3Ø)
      XIM=Ø.17D+39
      XAM=.ØDØ
      DO 1 I=Ø,N
```

Scanned with OKEN Scanner

```
      X=DABS(A(I))
      IF(X.LT.XIM.AND.X.NE.Ø.) XIM=X
      IF(X.GT.XAM) XAM=X
1     CONTINUE
      X=DSQRT(XIM*XAM)
      DO 2 I=Ø,N
2     A(I)=A(I)/X
      RETURN
      END
      SUBROUTINE EUCLID
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION U(Ø:3Ø),V(Ø:3Ø),Q(Ø:3Ø)
      COMMON/E1/U,N
      DO 1 I=Ø,N-1
1     V(I)=(N-I)*U(I)
      CALL WORMA(V,N-1)
      M=N-1
2     DO 3 K=Ø,N-M
      Q(K)=U(K)/V(Ø)
      DO 3 J=K+1,M+K
3     U(J)=U(J)-Q(K)*V(J-K)
      J=N-M+1
      DO 4 I=Ø,M-1
      Q(I)=U(J+I)
      I=-1
5     I=I+1
      IF(DABS(Q(I)).LE..1D-13.AND.I.LT.M-1) GOTO 5
      K=M-1-I
      IF(K.EQ.Ø.AND.DABS(Q(M-1)).LE..1D-13) GOTO 9
      IF(K.EQ.Ø) GOTO 11
      DO 7 I=Ø,M
```

Scanned with OKEN Scanner

```
7       U(I)=V(I)
        N=M
        J=M-1-K
        CALL WORMA(Q,M-1)
        DO 8 I=Ø,K
8       V(I)=Q(J+I)
        M=K
        GOTO 1
9       DO 1Ø I=Ø,M
1Ø      U(I)=V(I)
        N=M
        RETURN
11      N=Ø
        U(Ø)=1.DØ
        RETURN
        END
        SUBROUTINE SSTURM
        IMPLICIT DOUBLE PRECIZION(A-H,O-Z)
        DIMENSION U(Ø:3Ø),V(Ø:3Ø),Q(Ø:3Ø),S(Ø:3Ø,Ø:3Ø),N1(Ø:3Ø)
        COMMON/B1/U,N/B2/S,N1
        DO 1 I=Ø,N-1
        S(Ø,I)=U(I)
1       V(I)=(N-I)*U(I)
        S(Ø,N)=U(N)
        CALL WORMA(V,N-1)
        DO 1Ø I=Ø,N-1
1Ø      S(1,I)=V(I)
        N1(Ø)=N
        N1(1)=N-1
        DO 11 I=2,3Ø
        DO 11 K=Ø,3Ø
```

Scanned with OKEN Scanner

```
11      S(I,K)=.ØDØ
        DO 12 I=2,3Ø
12      N1(I)=-1
        M=N-1
        L=2
2       DO 3 K=Ø,N-M
        Q(K)=U(K)/V(Ø)
        DO 3 J=K+1,M+K
3       U(J)=U(J)-Q(K)*V(J-K)
        J=N-M+1
        DO 4 I=Ø,M-1
4       Q(I)=U(J+I)
        I=-1
5       I=I+1
        IF(DABS(Q(I)).LE..1D-13.AND.I.LT.M-1) GOTO 5
        K=M-1-I
        N1(L)=K
        DO 7 K=Ø,M
7       U(K)=V(K)
        N=M
        DO 8 K=I,M-1
        V(K-I)=-Q(K)
8       S(L,K-I)=-Q(K)
        M=N1(L)
        IF(L.GE.N1(Ø).OR.M.EQ.Ø) GOTO 9
        L=L+1
        GOTO 2
9       RETURN
        END
        SUBROUTINE ALG1
        IMPLICIT DOUBLE PRECISION(A-H,O-Z)
```

Scanned with OKEN Scanner

```
      DIMENSION U(Ø:3Ø)
      COMMON/E1/U,N/E3/A,B,NØ/E4/E
      Q=1.
      NA=KAV(A)
      NB=KAV(B)
      NØ=NA-NB
      IF(NØ.GT.Ø) GOTO 1
      A=B
      RETURN
1     X=U(Ø)
      DO 2 I=1,N
2     X=X*(A+Q)+U(I)
      IF(DABS(X).LE..1D-13) GOTO 4
      NAA=KAV(A+Q)
      IF(NAA-NB.LT.NØ) GOTO 3
      A=A+Q
      GOTO 1
3     Q=Q/1Ø.
      IF(Q.GE.E) GOTO 1
      Q=1.
      GOTO 5
4     Q=Q-E
      IF(Q.GE.E) GOTO 1
      Q=1.
5     X=U(Ø)
      DO 6 I=1,N
6     X=X*(B-Q)+U(I)
      IF(DABS(X).LE..1D-13) GOTO 8
      NBB=KAV(B-Q)
      IF(NA-NBB.LT.NØ) GOTO 7
      B=B-Q
```

Scanned with OKEN Scanner

```
      GOTO 5
7     Q=Q/10.
      IF(Q.GE.E) GOTO 5
      RETURN
8     Q=Q-E
      IF(Q.GE.E) GOTO 5
      RETURN
      END
      FUNCTION KAV(X)
      IMPLICIT DOUBLE PRECISION(A-H,O-Z)
      DIMENSION S(030,0:30),N1(030)
      COMMON/E2/S,N1
      KAV=0
      A=S(0,0)
      DO 1 I=1,N1(0)
1     A=A*X+S(0,I)
      J=1
3     B=S(J,0)
      DO 2 I=1,N1(J)
2     B=B*X+S(J,I)
      IF(A*B.LT.0.) KAV=KAV+1
      A=B
      J=J+1
      IF(N1(J).GE.0) GOTO 3
      RETURN
      END
```

Exemple rulaté :

```
I  DETERMINAREA RADACINILOR REALE A ECUATIEI P(X)=0
       PRECIZIA IMPUSA:0.00000000001
       COEFICIENTII POLINOMULUI
```

Scanned with OKEN Scanner

A(Ø)=1.Ø

A(1)=-5.Ø

A(2)=-8.9999999999999999

A(3)=64.999999999999994

A(4)=-15.999999999999996

A(5)=-178.99999999999996

A(6)= 138.9999999999999985

A(7)= -8.9999999999999999

A(8)= 64.999999999999994

A(9)= -15.999999999999996

A(1Ø)=-179.9999999999999985

A(11)= 143.9999999999999977

RADACINILE ECUATIEI

RADACINA REALA APROXIMATIVA:-2.999999999966476

RADACINA REALA APROXIMATIVA: Ø.9999999999981855

RADACINA REALA APROXIMATIVA:-1.000000000009140

RADACINA REALA APROXIMATIVA: 3.ØØØØØØØØØØ26180

RADACINA REALA APROXIMATIVA: 3.999999999993770

RADACINA REALA APROXIMATIVA:-2.ØØØØØØØØØØ24188

RADACINA REALA APROXIMATIVA: 2.ØØØØØØØØØØØ4Ø17

II.DETERMINAREA RADACINILOR REALE A ECUATIEI P(X)=Ø

PRECIZIA IMPUSA: Ø.ØØØØØØØØØØ1

COEFICIENTII POLINOMULUI

A(Ø)=-1.4142135623730000

A(1)= 1.7320508075680000

A(2)= 2.2360679774990000

A(3)= 3.1622776601680000

A(4)=12.599210498940000Ø

A(5)=-14.422495703Ø699998

A(6)= Ø.Ø

A(7)=-11.892Ø711500199997

A(8)=141.8765432112299995

A(9)= 69.3147180559899989

A(10)=-109.8162288659999991

A(11)= Ø.

A(12)= Ø.

A(13)= Ø.

A(14)=230.2585092939999945

A(15)= Ø.

A(16)= Ø.

A(17)= Ø.

A(18)=707.7657200209999874

A(19)=-3141.5926535799999427

RADACINILE ECUATIEI

RADACINA REALA APROXIMATIVA:-1.9014085526255443

RADACINA REALA APROXIMATIVA: 1.2504601133159088

RADACINA REALA APROXIMATIVA: 2.5779468983255369

1.4. Metode de tip Lehmer-Schur pentru determinarea zerourilor polinoamelor din $\mathbb{C}[z]$.

Vom prezenta în cele ce urmează algoritmul Lehmer-Schur şi o variantă îmbunătăţită a acestui algoritm.

Teorema 1.6. Coroana circulară $C(z_0,r,R)=\{c \mid c \in \mathbb{C} \quad r < |z_0 - c| < R,\ z_0 \in \mathbb{C},\ r$ şi $R \in \mathbb{R}_+\}$ poate fi acoperită complet cu discurile de rază

$$r_2 = \sqrt{r_1^2 - Rr} \ ,$$

unde $r_1 = (R+r)/2\cos(\alpha/2)$ şi reprezintă raza cercului pe care se află centrele discurilor de acoperire, iar α este unghiul de divizare (vezi fig.1) :

Scanned with OKEN Scanner

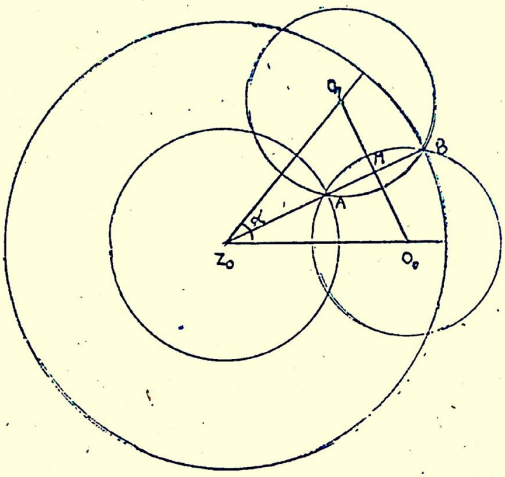


Fig.1.

Demonstrație.

$$|z_0A| = r \ , \quad |z_0B| = R \quad |z_0M| = \frac{r+R}{2} \quad |z_0O_0| = r_1 = (R+r)/2.\cos(\alpha/2)$$

$$|MO_0| = \frac{R+r}{2}\, tg(\frac{\alpha}{2}) \qquad |AO_0| = r_2 \qquad r_2 = \sqrt{r_1^2 - Rr}.$$

Corolarul 1. Deoarece unghiul α va trebui să fie un divizor al lui 360^0 fie $\alpha = 45^0$ atunci :

$$r_1 = (R+r)/\sqrt{2+\sqrt{2}} \quad \text{iar} \quad r_2 = \sqrt{R^2 - \sqrt{2}Rr + r^2}/\sqrt{2+\sqrt{2}}$$

iar centrele celor 8 discuri de acoperire vor fi :

$$C_j = z_0 + r_2 \cdot e^{i(j.\frac{\pi}{4})} \quad \text{cu} \quad j = \overline{0,7} \ , \ \text{și} \quad i^2 = -1.$$

Corolarul 2. Dacă $R = 2r$ atunci :

$$r_1 = 3r/\sqrt{2+\sqrt{2}} \qquad r_2 = r_1\sqrt{7-9\sqrt{2}/2}$$

Observația 1.7. Este cunoscut că toate zerourile polinomului (1.1) se află în coroana circulară determinată de :

$$|a_n|/(A+|a_n|) < |z_i^*| < 1 + B/|a_0| \ , \ i = \overline{1,n} \quad (1.10)$$

cu $A = \max\{|a_0|, |a_1|, \ldots, |a_{n-1}|\}$ şi $B = \max\{|a_1|, |a_2|, \ldots, |a_n|\}$.

Algoritmul 1.4. (Lehmer-Schur) [37(1), 41(1)].

Considerăm polinomul P (1.1). Fie $z_0 = (0.,0.)$, $r_0 = 1$ şi discul $D(z_0,r_0) = \{c \in \mathbb{C} \mid |z_0 - c| < r_0\}$. Notăm cu , $r = 2^k.r_0$ unde $k \in Z$ a cărui valoare iniţială este 0.

Pasul 1. - Dacă polinomul P nu are zerouri în $D(z_0,r_0)$ se execută pasul 2.

- Dacă polinomul P are zerouri în $D(z_0,r_0)$ se execută pasul 3.

Pasul 2. Facem $k = k+1$ şi considerăm discul $D(z_0,r)$ pînă găsim discul în care polinomul P are zerouri. Fie $r = 2^{k_0+1}r_0$ raza discului căutat. Deoarece în discul de rază $r = 2^{k_0}r_0$ P nu a avut zerouri rezultă că P are zerouri în coroana circulară de lăţime $r_1 = 2^{k_0}r_0$:

$$C(z_0,r_1,2r_1) = \{c \in \mathbb{C} \mid r_1 \leq |c-z_0| \leq 2r_1\}$$

Se continuă cu pasul 4.

Pasul 3. Facem $k = k-1$, considerăm discurile $D(z_0,r)$ pînă găsim discul în care polinomul P nu are zerouri. Fie $r = 2^{k_0}r_0$ raza discului căutat. Deoarece în discul de rază $r = 2^{k_0+1}.r_0$ P a avut încă zerouri rezultă că P are zerouri în coroana circulară de lăţime $r_1 = 2^{k_0}r_0$.

$$C(z_0,r_1,2r_1) = \{c \in \mathbb{C} \mid r_1 \leq |c-z_0| \leq 2r_1\}$$

Pasul 4. Coroana circulară $C(z_0,r_1,2r_1)$ poate fi acoperită cu 8 discuri

$$S(c_j,r_2) = \{c \in \mathbb{C} \mid |c-c_j| \leq r_2\}$$

unde $c_j = z_0 + 3r_1[\cos(j\pi/4) + i\sin(j\pi/4)]/\sqrt{2+\sqrt{2}}$ $j = \overline{0,7}$ şi $r_2 = r_1\sqrt{7-9\sqrt{2}/2}$. Căutăm în care din aceste discuri $S(c_j,r_2)$, P are zerouri. Fie mulţimea : $J = \{j \mid j \in \{0,1,\ldots,7\}$, pentru care P are zerouri în $S(c_j,r_2)$ şi $j-1 \notin J\}$. Mulţimea $J \neq \emptyset$.

Scanned with OKEN Scanner

- Dacă $r_2 < \varepsilon$ (precizia impusă) atunci am determinat zeroul c_j, $j \in J$; Continuăm cu pasul 5.

- Dacă $r_2 \geq \varepsilon$, atunci fie $r_0 = r_2$ şi $z_0 = c_j$ pentru primul $j \in J$ şi se reia algoritmul de la pasul 1.

Pasul 5. Aplicînd schema lui Horner pe rînd pentru fiecare zerou aproximativ c_j, $j \in J$ determinat la pasul 4 va rezulta un polinom P_1 al cărui grad va fi cel puţin cu unu mai mic ca al lui P. Cu polinomul P_1 se reia algoritmul avînd ca disc iniţial pe $S(c_j, r_2)$, unde j este primul indice din J. Dacă gradul polinomului P_1 este ≤ 2 vom folosi după caz formula pentru determinarea zerourilor polinoamelor de grad ≤ 2.

Convergenţa metodei

Vom determina convergenţa metodei pentru cazul cel mai puţin favorabil şi pentru "primul zerou" găsit al polinomului P. Raţionamentul se poate relua pentru "primul zerou" întîlnit al polinomului P_1. Presupunem că "primul zerou" întîlnit al polinomului P nu este în discul $S(z_0, r_0)$, atunci se execută pasul 2. Fie k_0+1, unde $k_0 \in \mathbb{N}$, astfel încît primul zerou întîlnit să se afle în coroana circulară de lăţime $r_1^0 = 2^{k_0} r_0$. In pasul 4 care urmează a fi executat se determină discul de rază $r_2^0 = \sigma r_1^0$ (unde $\sigma = \sqrt{7-9\sqrt{2}/2} \cong 0.797...$) care conţine "primul zerou întîlnit de algoritm" al lui P. In continuare pînă ce $r_2 > \varepsilon$, algoritmul va executa ciclic pasul 3 şi pasul 4. In pasul 4 dacă eroarea nu este realizată atunci se face $r_0^1 = r_2^0$ iar în pasul 3 se va determina $k_1 \in \mathbb{Z}_-$ (în cel mai rău caz $k_1 = -1$) astfel încît zeroul să se afle în coroana circulară de lăţime $r_1^1 = 2^{k_1} r_0^1$. In continuare în pasul 4 se va determina în care din cele 8 discuri de rază $r_2^1 = \sigma r_1^1$ se află zeroul ... Aşa rezultă şirul razelor discurilor în care se află "primul zerou întîlnit de algoritm" al polinomului P.

$$r_2^i = \sigma^{i+1} \cdot 2^{k_0+k_1+\ldots+k_i} \cdot r_0 \qquad (1.11)$$

unde $k_0 \in \mathbb{Z}_+$ şi $k_1, k_2, \ldots, k_i \in \mathbb{Z}_-$ iar $\sigma = \sqrt{7-9\sqrt{2}/2} \cong 0.797\ldots$ şi care $r_2^i \longrightarrow 0$ evident cînd $i \longrightarrow \infty$.

Observaţia 1.8. Dacă $T^m(Q(0)) = 0$, $T^k(Q(0)) > 0$ pentru $\forall\, k = \overline{1,m-1}$ şi $T^m(Q(z)) \neq$ constantă, teorema nu precizează nimic (vezi Observaţia 1.5)). Pentru a evita acest caz în cadrul programului prezentat mai jos s-a procedat în felul următor : Deoarece coeficienţii polinomului Q depind de r_0 şi o vom aduna la r cantitatea ε (precizia impusă) în plus sau în minus după cum sîntem la pasul 2 sau 3 şi aplicăm criteriul de existenţă a zerourilor (teorema 1.4) în acest disc.

Observaţia 1.9. Prin aplicarea teoremei 1.4 şi calculul polinomului Q se pot genera în cazul polinoamelor de grad mare, coeficienţi foarte mari sau foarte mici în valoare absolută, ceea ce poate provoca depăşirea capacităţii de lucru a calculatorului. Pentru aceasta se recomandă la fiecare pas al teoremei 1.4 şi la fiecare compunere (observaţia 1.5) să se normalizeze coeficienţii obţinuţi.

Observaţia 1.10. In program s-a preferat să se facă iniţializările $z_0 = (0.1, 0.)$ şi $r_0 = |a_n|/(1 + |a_n|)$, formula (1.10) pentru a evita paşii inutili la pornirea algoritmului.

Observaţia 1.11. Algoritmul are următoarele avantaje : convergenţa metodei nu este restricţionată de condiţii iniţiale şi determină toate zerourile polinomului (1.1). Printre dezavantajele metodei amintim : o viteză de convergenţă liniară; nu se pot obţine precizii mari în determinarea zerourilor polinomului (1.1). De asemeni dacă pentru "primul zerou" determinat se poate spune că a fost determinat cu precizia ε , despre zerourile determinate mai apoi nu mai poate fi afirmat că sînt zerouri determinate

Scanned with OKEN Scanner

```
      IF(N.EQ.1) GOTO 34
      STOP
33    ZØ=(-A(2)+CSQRT(A(2)*A(2)-4.*A(1)*A(3)))/(2.*A(1))
      K7=K7+1
      IF(DA.EQ.'NU') GOTO 42
      WRITE(2,28) K7,ZØ
42    WRITE(1,28) K7,ZØ
      ZØ=(-A(2)-CSQRT(A(2)*A(2)-4.*A(1)*A(3)))/(2.*A(1))
      K7=K7+1
      IF(DA.EQ.'NU') GOTO 43
      WRITE(2,28) K7,ZØ
43    WRITE(1,28) K7,ZØ
      STOP
34    ZØ=-A(2)/A(1)
      K7=K7+1
      IF(DA.EQ.'NU') GOTO 44
      WRITE(2,28) K7,ZØ
      WRITE(1,28) K7,ZØ
      STOP
96    WRITE(1,86)
86    FORMAT(1X,'EROARE IN TABLOUL E.')
      STOP
97    WRITE(1,87)
87    FORMAT(1X,'EROARE LA DETERMINAREA COROANEI',
     1'CIRCULARE.STOP.')
      STOP
98    WRITE(1,88)
88    FORMAT(1X,'VALOAREA ABSOLUTA A SOLUTIILOR  1.E-7'
     1'SAU  1.E+7;ALEGE UN ALT PUNCT INITIAL')
1ØØ   STOP
      END
```

Scanned with OKEN Scanner

```
      YC=(Q(I)/Q(Ø))**(1./I)
      IF(YC.GT.RN) RN=YC
6     CONTINUE
      RN=2.*RN
77    Q1=RN*Q(Ø)
      DO 75 I=1,N-1
75    Q1=(Q1-Q(I))*RN
      Q1=Q1-Q(N)
      Q11=RN*N*Q(Ø)
      DO 76 I=1,N-2
76    Q11=(Q11-(N-I)*Q(I))*RN
      Q11=Q11-Q(N-1)
      QQ=Q1/Q11
      IF(QQ.LT.E) GOTO 78
      RN=RN-QQ
      GOTO 77
78    RN=RN-QQ
      RM=RN
85    Q2Ø=RM*Q(Ø)
      DO 79 I=1,N-1
79    Q2Ø=(Q2Ø+Q(I))*RM
      Q2Ø=Q2Ø-Q(N)
      IF(Q2Ø.LT.Ø.) GOTO 86
      Q22=Q2Ø
      RM=RM/2.
      GOTO 85
86    Q2Ø=Q22
      X1=RM*Q(N)/(Q(N)+Q2Ø)
83    Q2X=X1*Q(Ø)
      DO 8Ø I=1,N-1
8Ø    Q2X=(Q2X+Q(I))*X1
```

Scanned with OKEN Scanner

```
      Q2X = Q2X-Q(N)
      XØ=(RM*Q2X-X1*Q2Ø)/(Q2X-Q2Ø)
      IF(XØ-X1,LT.E) GOTO 82
      X1=XØ
      GOTO 83
82    RM=XØ
      D(2)=(Ø.,Ø.)
      EH=Ø.1
61    D(1)=(1.,Ø.)*(RM+EH)
      CALL COMPUN
      CALL SCHUR
      IF(KOD.EQ.Ø.OR.KOD.EQ.2) GOTO 62
      RM=RM+EH
      GOTO 61
62    EH=EH/1Ø.
      IF(EH.GE.Ø.ØØ1) GOTO 61
      DO 63 I=Ø,N
63    UU(I)=U(I)
      DO 64 I=Ø,N
64    U(I)=UU(N-I)
      EH=Ø.1
      RQ=1./RN
65    D(1)=(1.,Ø.)*(RQ+EH)
      CALL COMPUN
      CALL SCHUR
      IF(KOD.EQ.Ø.OR.KOD.EQ.2) GOTO 66
      RQ=RQ+EH
      GOTO 65
66    EH=EH/1Ø.
      IF(EH.GE.Ø.ØØØØ1) GOTO 65
      RN=1./RQ
```

Scanned with OKEN Scanner

```
      DO 67 I=Ø,N
67    U(I)=UU(I)
      WRITE(1,84) RN,RM
      WRITE(2,84) RN,RM
84    FORMAT(1X,'RAZA MARE A COROANEI=',G/
     11X,'RAZA MICA A COROANEI=',G//)
      L=Ø
      C2=SQRT(2.+SQRT(2.))
      R1=(RM+RN)/C2
      R2=SQRT(R1*R1-RM*RN)
      Z=(Ø.,.78539816)
      D(1)=(1.,Ø.)*R2
      DO 8 J=Ø,7
      D(2)=R1*CEXP(Z*J)
2     CALL COMPUN
      CALL SCHUR
      IF(KOD.NE.Ø) GOTO 1
      L=L+1
      O(L)=D(2)
      R(L)=R2
      GOTO 8
1     IF(KOD.EQ.1) GOTO 8
      IF(KOD.EQ.3) GOTO 1ØØ
      R2=R2+E
      D(1)=(1.,Ø.)*R2
      GOTO 2
8     CONTINUE
      LL=L
9     MG=Ø
      DO 15 I=1,L
      I1=Ø
```

Scanned with OKEN Scanner

```
      IF(R(I).LT.E) GOTO 15
      RØ=R(I)/2.
1Ø    D(1)=(1.,Ø.)*RØ
      D(2)=O(I)
4     CALL COMPUN
      CALL SCHUR
      IF(KOD.NE.Ø) GOTO 3
      IF(RØ.LT.E) GOTO 21
      RØ=RØ/2.
      GOTO 1Ø
21    O(I)=D(2)
      R(I)=RØ
      I1=1
      GOTO 11
3     IF(KOD.EQ.1) GOTO 11
      IF(KOD.EQ.3) GOTO 1ØØ
      RØ=RØ+E
      D(1)=(1.,Ø.)*RØ
      GOTO 4
11    R1=(R(I)+RØ)/C2
      R2=SQRT(R1*R1-R(I)*RØ)
      OO=O(I)
      D(1)=(1.,Ø.)*R2
      DO 14 J=Ø,7
      D(2)=R1*CEXP(Z*J)+OO
7     CALL COMPUN
      CALL SCHUR
      IF(KOD.NE.Ø) GOTO 5
      IF(I1.NE.Ø) GOTO 12
      O(I)=D(2)
      R(I)=R2
```

Scanned with OKEN Scanner

```
      IF(R2.GE.E) MG=1
      I1=1
      GOTO 13
12    IF(LL.GE.1ØØØ) GOTO 19
      LL=LL+1
      O(LL)=D(2)
      R(LL)=R2
13    IF(R2.GE.E) MG=1
      GOTO 14
5     IF(KOD.EQ.1) GOTO 14
      IF(KOD.EQ.3) GOTO 1ØØ
      RØ=RØ+E
      D(1)=(1.,Ø.)*RØ
      GOTO 7
14    CONTINUE
15    CONTINUE
      L=LL
      IF(MG.EQ.1) GOTO 9
      K1=Ø
      DO 16 I=1,L-1
      IF(O(I).EQ.(Ø.,Ø.)) GOTO 16
      K=I
      DO 17 J=I+1,L
      IF(O(J).EQ.(Ø.,Ø.)) GOTO 17
      IF(CABS(O(K)-O(J).GT.R(K)+R(J)) GOTO 17
      IF(R(K).LE.R(J)) GOTO 18
      O(K)=(Ø.,Ø.)
      K=J
      GOTO 17
18    O(J)=(Ø.,Ø.)
17    CONTINUE
      K1=K1+1
```

```
      IF(NA.EQ.Ø) GOTO 45
      WRITE(2,88) K1,O(K),R(K)
88    FORMAT(1X,'Z(',I2,') = (',G,',',G,')'/
     19X,'CU FRECIZIA ',G)
45    WRITE(1,81)K1,O(K),R(K)
81    FORMAT(1X,'Z(',I2,') = (',G,',',G,') ',
     1'CU PRECIZIA ',G)
      O(K)=(Ø.,Ø.)
16    CONTINUE
      IF(K1.GE.N) GOTO 91
      WRITE(1,89)
89    FORMAT(1X,'AVEM ZEROURI LA DISTANTE MAI MICI CA ',
     1'PRECIZIA IMPUSA')
      RETURN
91    IF(K1.GT.N) GOTO 92
      RETURN
92    WRITE(1,93)
93    FORMAT(1X,'EROARE MAI MULTE ZEROURI DECIT GRADUL ',
     1'POLINOMULUI')
      STOP
19    WRITE(1,2Ø)
2Ø    FORMAT(1X,'DIMENSIUNEA TABLOURILOR O SI R',
     1'PREA MICA. STOP')
      STOP
1ØØ   WRITE(1,22)
22    FORMAT(1X,'EROARE GRAVA IN SUBPROGRAMUL SCHUR.',
     1' STOP')
      STOP
      END
      SUBROUTINE COMPUN
      COMPLEX U,B,D,D1,D2
```

Scanned with OKEN Scanner

```
      DIMENSION U(Ø:15),B(Ø:15),D(2),C(Ø:15,Ø:15)
      COMMON/E1/U/E3/N/E5/C/E6/D/E7/B
      DO 1 I=Ø,N
      B(I)=(Ø.,Ø.)
      D1=(1.,Ø.)
      DO 2 L=1,N-I
2     D1=D1*D(1)
      DO 1 K=Ø,I
      D2=(1.,Ø.)
      DO 3 L=1,I-K
3     D2=D2*D(2)
1     B(I)=B(I)+C(N-K,I-K)*U(K)*D1*D2
      XIM=.17E+39
      XAM=Ø.
      DO 13 I=Ø,N
      X=CABS(B(I))
      IF(X.LT.XIM.AND.X.NE.Ø.) XIM=X
      IF(X.GT.XAM) XAM=X
13    CONTINUE
      Y=SQRT(XIM)*SQRT(XAM)
      DO 14 I=Ø,N
14    B(I)=B(I)/Y
      RETURN
      END
      SUBROUTINE SCHUR
      COMPLEX A,B
      DIMENSION A(Ø:15),B(Ø:15)
      COMMON/E3/N/E7/B/E8/KOD
      IF(B(N).EQ.(Ø.,Ø.)) GOTO 1Ø
      M=1
      K=N
```

Scanned with OKEN Scanner

```
      L=1
      KM=Ø
8     MT1=L
      L=Ø
      DO 2 I=Ø,K-1
      A(I)=CONJG(B(K))*B(I)-B(Ø)*CONJG(B(K-I))
      IF(A(I).NE.(Ø.,Ø.)) L=1
2     CONTINUE
      A(K)=CONJG(B(K))*B(K)-B(Ø)*CONJG(B(Ø))
      XIM=.17E+39
      XAM=Ø.
      DO 9 I=1,K
      X=CABS(A(I))
      IF(X.LT.XIM.AND.X.NE.Ø.) XIM=X
      IF(X.GT.XAM) XAM=X
9     CONTINUE
      Y=SQRT(XIM)*SQRT(XAM)
      IF(REAL(A(K)).EQ.Ø.) GOTO 6
      IF(REAL(A(K)).LT.Ø.) KM=M
      B(K)=(Ø.,Ø.)
      DO 7 I=Ø,K-1
      IF(Y.NE.Ø.) A(I+1)=A(I+1)/Y
7     B(I)=A(I+1)
      M=M+1
      K=K-1
      IF(K.GE.1) GOTO 3
      MT1=Ø
6     IF(KM.NE.Ø) GOTO 11
      IF(MT1.EQ.Ø) GOTO 12
      KOD=2
      RETURN
```

Scanned with OKEN Scanner

```
11    KOD=Ø
      RETURN
12    KOD=1
      RETURN
10    WRITE(1,21)
21    FORMAT(1X,'P(Ø)=Ø EROARE')
      KOD=3
      RETURN
      END
```

Exemplu :

```
REZOLVAREA ECUATIEI P(Ź)=Ø
    PRECIZIA IMPUSA: Ø.1
    COEFICIENTII POLINOMULUI
       A(Ø) = (1.,Ø.)
       A(1) = (4.,-12.)
       A(2) = (-46.,-48.)
       A(3) = (-124.,2ØØ.)
       A(4) = (129.,8Ø.)
       A(5) = (-624.,5812.)
       A(6) = (11796.,158Ø8.)
       A(7) = (29184.,-2184Ø.)
       A(8) = (-4Ø72Ø.,Ø.)
    RĂDĂCINILE ECUATIEI
Z(1)=(4.992222,-Ø.129533E-Ø1)  CU PRECIZIA  Ø.542ØØ49E-Ø1
Z(2)=(-Ø.65Ø925E-Ø6, 6.Ø2Ø199) CU PRECIZIA  Ø.5999981E-Ø1
Z(3)=(Ø.1654791E-Ø1, 8.ØØ4621) CU PRECIZIA  Ø.4ØØ9113E-Ø1
Z(4)=(-2.975621, Ø.1711624E-Ø1)CU PRECIZIA  Ø.4ØØ9113E-Ø1
Z(5)=(1.ØØ8834, Ø.322396E-Ø1)  CU PRECIZIA  Ø.4ØØ911ØE-Ø1
Z(6)=(-7.Ø28688,Ø.598Ø3Ø5E-Ø2) CU PRECIZIA  Ø.3216881E-Ø1
Z(7)=(-Ø.4436403E-Ø2,1.983613) CU PRECIZIA  Ø.2485273E-Ø1
Z(8)=(Ø.197Ø6Ø4E-Ø1,-3.999439) CU PRECIZIA  Ø.4Ø336Ø5E-Ø1
```

Scanned with OKEN Scanner

1.5. Metode cu ordine superioare de convergenţă de determinare simultană a zerourilor polinoamelor algebrice

Prezentăm metode cu ordine de convergenţă superioare, care determină simultan zerourile polinoamelor algebrice. Aceste metode le putem folosi pentru a îmbunătăţi precizia zerourilor "localizate" cu ajutorul metodelor de tip "căutare". Metoda Durand-Kerner.

Teorema 1.7. Fie $T: C^k \longrightarrow C^k$ un operator de p ori continuu diferenţiabil şi $x^* \in C^k$ un punct fix a lui T. Presupunem că $T'(x^*) = \ldots = T^{(p-1)}(x^*) = 0$. Atunci x^* este punct de atracţie pentru iteraţia $x^{k+1} = Tx^k$ iar ordinul de convergenţă este cel puţin p. [41(1)].

Fie P polinomul (1.1) cu zerourile $z_1^*, z_2^*, \ldots, z_n^*$, distincte. Polinomul P se mai poate scrie $P(z) = a_0 \prod_{i=1}^{n} (z - z_i^*)$. Considerăm că $z_1^k, z_2^k, \ldots, z_n^k$ sînt zerourile aproximative ale polinomului P, adică $z_i^k \longrightarrow z_i^*$ pentru orice $i = \overline{1,n}$ cînd $k \longrightarrow \infty$. Dacă considerăm polinoamele $Q_k(z) = a_0 \prod_{i=1}^{n} (z - z_i^k)$, atunci cînd $k \longrightarrow \infty$ rezultă că $Q_k(z) \longrightarrow P(z)$ şi $Q_k'(z) \longrightarrow P'(z)$. Metoda Durand-Kerner foloseşte metoda Newton în care $P'(z)$ se înlocuieşte cu $Q_k'(z)$. Notăm cu T operatorul :

$$T(Z^k) = \begin{bmatrix} z_1^k - P(z_1^k)/Q_k'(z_1^k) \\ z_2^k - P(z_2^k)/Q_k'(z_2^k) \\ \vdots \\ z_n^k - P(z_n^k)/Q_k'(z_n^k) \end{bmatrix} \qquad (1.12)$$

unde $Q_k'(z_i^k) = a_0 \prod_{\substack{j=1 \\ j \neq i}}^{n} (z_i^k - z_j^k)$, iar $Z^k = (z_1^k, z_2^k, \ldots, z_n^k)^T$. Putem folosi procesul iterativ

$$Z^{k+1} = T(Z^k) \qquad (1.13)$$

Deoarece Z^* nu are componente identice rezultă că $Q_k'(z_i^*) \neq 0$ pentru $i = \overline{1,n}$ şi $Q_k'(z_i^k) \neq 0$ pentru $i = \overline{1,n}$ pe o anumită

sferă centrată în Z^*. Operatorul T este continuu diferențiabil pe o sferă centrată în Z^* pentru că P și Q'_k sînt polinoame. Evident că $T(Z^*) = Z^*$ deci Z^* este punct fix pentru operatorul T. Calculăm $T'(Z^*)$

$$T'(Z^*) = \begin{bmatrix} 1 - (P'(z_1^*).P'(z_1^*) - P(z_1^*).P''(z_1^*))/(P'(z_1^*))^2 \\ 1 - (P'(z_2^*).P'(z_2^*) - P(z_2^*).P''(z_2^*))/(P'(z_2^*))^2 \\ \vdots \\ 1 - (P'(z_n^*).P'(z_n^*) - P(z_n^*).P''(z_n^*))/(P'(z_n^*))^2 \end{bmatrix}$$

Evident $T'(Z^*) = 0$. Conform teoremei 1.7 formulele 1.12 determină un proces iterativ convergent la Z^*, iar ordinul de convergență este cel puțin 2.

Programul DURAND materializează algoritmul prezentat mai sus :

```
      PROGRAM DURAND
      COMPLEX A,Z,Q,P,Q1
      COMMON/ET1/A(0:25)/ET2/Z(25)/ET3/N
      CALL ASSIGN(1,'SY:D.DAT')
      CALL ASSIGN(2,'SY:D.LST')
      READ(1,101)N,E
101   FORMAT(I5,F7.6)
      READ(1,102) (A(I),I=0,N)
102   FORMAT(6F)
      READ(1,102) (Z(I),I=1,N)
      WRITE(2,103) E,(I,A(I),I=0,N)
103   FORMAT(1X,'REZOLVAREA ECUATIEI P(Z)=0 CU METODA DURAND',
     1'-KERNER'/5X,'PRECIZIA IMPUSA:',F/
     25X,'COEFICIENTII POLINOMULUI'/
     3<N+1>(1X,'A(',I2,')=(',F,',',F,')'/),
     41X,'RADACINILE APROXIMATIVE ALE ECUATIEI')
      WRITE(2,104) (I,Z(I),I=1,N)
```

```
      WRITE(2,1Ø5)
1Ø5   FORMAT(1X,'RADACINILE CALCULATE CU METODA DURAND-KERNER')
      K=Ø
1     E1=Ø.
      K=K+1
      DO 2 I=1,N
      Q=P(Z(I))/Q1(Z(I))
      E2=CABS(Q)
      IF(E2.GT.E1) E1=E2
2     Z(I)=Z(I)-Q
      IF(E1.GE.E) GOTO 1
      WRITE(2,1Ø6) K
1Ø6   FORMAT(1X,'NUMAR ITERATII : ',I2)
      WRITE(2,1Ø4) (I,Z(I),I=1,N)
1Ø4   FORMAT(<N>(1X,'Z(',I2,')=(',F15.7,',',F15.7,')'/))
      STOP
      END
      COMPLEX FUNCTION P(Z)
      COMPLEX A,Z
      COMMON/ET1/A(Ø:25)/ET3/N
      P=A(Ø)
      DO 1 I=1,N
1     P=P*Z+A(I)
      RETURN
      END
      COMPLEX FUNCTION Q1(X)
      COMPLEX X,Z
      COMMON/ET2/Z(25)/ET3/N
      Q1=(1.,Ø.)
      DO 1 I=1,N
      IF(Z(I).EQ.X) GOTO 1
```

Scanned with OKEN Scanner

```
      Ql=Ql*(X-Z(I))
1     CONTINUE
      RETURN
      END
```

Exemplu rulat.

```
REZOLVAREA ECUATIEI P(Z)=Ø CU METODA DURAND-KERNER
      PRECIZIA IMPUSA: Ø.ØØØØØ1
      COEFICIENTII POLINOMULUI
        A(Ø) = (     1.ØØØØØØØ,      Ø.ØØØØØØØ)
        A(1) = (     4.ØØØØØØØ,     12.ØØØØØØØ)
        A(2) = (   -46.ØØØØØØØ,    -48.ØØØØØØØ)
        A(3) = (  -124.ØØØØØØØ,    2ØØ.ØØØØØØØ)
        A(4) = (   129.ØØØØØØØ,    8Ø.ØØØØØØØ)
        A(5) = (  -624.ØØØØØØØ,   5812.ØØØØØØØ)
        A(6) = ( 11796.ØØØØØØØ,  158Ø8.ØØØØØØØ)
        A(7) = ( 29184.ØØØØØØØ, -2184Ø.ØØØØØØØ)
        A(8) = (-4Ø32Ø.ØØØØØØØ,      Ø.ØØØØØØØ)
      RADACINILE APROXIMATIVE ALE ECUATIEI
        Z(1) = (     Ø.9927228,      Ø.ØØØØØØØ)
        Z(2) = (     Ø.ØØ49455,      1.9998536)
        Z(3) = (    -Ø.ØØ67976,      5.9965428)
        Z(4) = (    -Ø.ØØ67977,      7.9961418)
        Z(5) = (    -7.Ø1Ø67Ø2,      Ø.ØØ29347)
        Z(6) = (     2.9653745,     -Ø.Ø287132)
        Z(7) = (     4.98Ø752Ø,      Ø.Ø131138)
        Z(8) = (     Ø.1122128,      3.9798788)
      RADACINILE CALCULATE CU METODA DURAND-KERNER
      NUMAR ITERATII: 4
        Z(1) = (     1.ØØØØØØØ,      Ø.ØØØØØØØ)
        Z(2) = (     Ø.ØØØØØØØ,      2.ØØØØØØØ)
```

Z(3) = (-0.0000001, 6.0000000)
Z(4) = (0.0000004, 8.0000000)
Z(5) = (-7.0000000, 0.0000001)
Z(6) = (-3.0000000, 0.0000001)
Z(7) = (5.0000000, 0.0000000)
Z(8) = (0.0000001, -4.0000000)

Metoda Ehrlich

Fie polinomul (1.1) cu zerourile $z_1^*, z_2^*, \ldots, z_n^*$ distincte. Polinomul P se mai poate scrie $P(z) = a_0 \prod_{i=1}^{n} (z - z_i^*)$. Considerăm că $z_1^k, z_2^k, \ldots, z_n^k$ sînt zerourile aproximative ale polinomului P, adică $z_i^k \longrightarrow z_i^*$ pentru $i = \overline{1,n}$ cînd $k \longrightarrow \infty$. Fie polinomul $Q_k(z) = a_0 \prod_{i=1}^{n} (z - z_i^k)$. Cînd $k \longrightarrow \infty$ avem $Q_k(z) \longrightarrow P(z)$ şi $Q_k'(z) \longrightarrow P'(z)$. Evident că polinomul $P(z_i^*)/Q_k'(z_i^k)$ aproximează pe $z_i^k - z_i^*$. Ideea metodei Ehrlich este de a aplica metoda lui Newton funcţiei $P(z_i^k)/Q_k'(z_i^k)$. Rezultă procesul iterativ :

$$z_1^{k+1} = z_1^k - \frac{P(z_1^k)}{P'(z_1^k)} \left[1 - \frac{P(z_1^k)}{P'(z_1^k)} \sigma_1^k \right]^{-1}$$

$$\vdots \qquad \vdots \qquad \vdots$$

$$z_n^{k+1} = z_n^k - \frac{P(z_n^k)}{P'(z_n^k)} \left[1 - \frac{P(z_n^k)}{P'(z_n^k)} \sigma_n^k \right]^{-1}$$

unde $\sigma_i^k = \sum_{\substack{j=1 \\ j \neq i}}^{n} (z_j^k - z_i^k)^{-1}$. Convergenţa metodei se demonstrează cu ajutorul teoremei 1.7 [41(1)], iar ordinul de convergenţă este cel puţin egal cu 3. Acest algoritm este folosit în programul EHRLICH.

Observaţia 1.14. Condiţiile de convergenţă a metodelor mai sus prezentate sînt greu de verificat, dar dacă pentru $Z^0 = (z_1^0, z_2^0, \ldots, z_n^0)$ metoda aleasă nu converge, adică programul

nu se termină într-un număr relativ mic de iteraţii (< 10) rezultă că nu sîntem în sfera de convergenţă a metodei şi deci va trebui să determinăm cu metoda LEHMER o soluţie iniţială Z^0 mai bună şi să reluăm algoritmul ales.

Observaţia 1.15. In mod analog se poate folosi metoda lui Newton

$$z_1^{k+1} = z_1^k - P(z_1^k)/P'(z_1^k)$$
$$\vdots \qquad \vdots$$
$$z_n^{k+1} = z_n^k - P(z_n^k)/P'(z_n^k)$$

cu ordin de convergenţă 2, sau metode tip Cebîşev cu ordinul de convergenţă 3 :

$$z_1^{k+1} = z_1^k - P(z_1^k)/P'(z_1^k) - P''(z_1^k)\cdot(P(z_1^k))^2/(2\cdot P'(z_1^k))^3$$
$$\vdots \qquad \vdots$$
$$z_n^{k+1} = z_n^{k+1} - P(z_n^k)/P'(z_n^k) - P''(z_n^k)\cdot(P(z_n^k))^2/(2\cdot P'(z_n^k))^3$$

Convergenţa acestor metode se demonstrează cu ajutorul teoremei 1.7 în mod asemănător cu convergenţa metodei Durand-Kerner.

```
      PROGRAM EHRLICH
      COMPLEX A,Z,P,P1,S,Q
      COMMON/ET1/A(0:25)/ET2/Z(25)/ET3/N
      CALL ASSIGN(1,'SY:E.DAT')
      CALL ASSIGN(2,'SY:E.LST')
      READ(1,101) N,E
  101 FORMAT(I5,F7.6)
      READ(1,102) (A(I), I=0,N)
  102 FORMAT(6F)
      READ(1,102) (Z(I),I=1,N)
      WRITE(2,103) E,(I,A(I),I=0,N)
  103 FORMAT(1X,'REZOLVAREA ECUATIEI P(Z)=0 CU METODA EHRLICH'/
```

```
      15X,'PRECIZIA IMPUSA:',F/
      25X,'COEFICIENTII POLINOMULUI*/
      3<N+1>(1X,'A(',I2,')=(',F,',',F,')'/)
      41X,'RADACINILE APROXIMATIVE ALE ECUATIEI')
      WRITE(2,104) (I,Z(I),I=1,N)
104   FORMAT(<N>(1X,'Z(',I2,')=(',F15.7,',',F15.7,')'/))
      WRITE(2,105)
105   FORMAT(1X,'RADACINILE CALCULATE CU METODA EHRLICH')
      K=0.
1     E1=0.
      K=K+1
      DO 2 I=1,N
      Q=P(Z(I))/P1(Z(I))
      Q=Q/((1.,0.)-Q*S(I))
      E2=CABS(Q)
      IF(E2.GT.E1) E1=E2
2     Z(I)=Z(I)-Q
      IF(E1.GE.E) GOTO 1
      WRITE(2,106) K
106   FORMAT(1X,'NUMAR ITERATII :',I2)
      WRITE(2,104) (I,Z(I),I=1,N)
      STOP
      END
      COMPLEX FUNCTION P(Z)
      COMPLEX A,Z
      COMMON/ET1/A(0:25)/ET3/N
      P=A(0)
      DO 1 I=1,N
1     P=P*Z+A(I)
      RETURN
      END
```

Scanned with OKEN Scanner

```
      COMPLEX FUNCTION P1(Z)
      COMPLEX A,Z
      COMMON/ET1/A(Ø:25)/ET3/N
      P1=N*A(Ø)
      DO 1 I=1,N-1
1     P1=P1*Z+(N-I)*A(I)
      RETURN
      END
      COMPLEX FUNCTION S(I)
      COMPLEX Z
      COMMON/ET2/Z(25)/ET3/N
      S=(Ø.,Ø.)
      DO 1 J=1,N
      IF(J.EQ.I) GOTO 1
      S=S+(1.,Ø.)/(Z(I)-Z(J))
1     CONTINUE
      RETURN
      END
```

Exemplu rulat.

```
      REZOLVAREA ECUATIEI P(Z)=Ø CU METODA EHRLICH
         PRECIZIA IMPUSA: Ø.ØØØØØ1
         COEFICIENTII POLINOMULUI
      A(Ø)= (      1.ØØØØØØØ,       Ø.ØØØØØØØ)
      A(1)= (      4.ØØØØØØØ,     -12.ØØØØØØØ)
      A(2)= (    -46.ØØØØØØØ,     -48.ØØØØØØØ)
      A(3)= (   -124.ØØØØØØØ,     2ØØ.ØØØØØØØ)
      A(4)= (    129.ØØØØØØØ,      8Ø.ØØØØØØØ)
      A(5)= (   -624.ØØØØØØØ,    5812.ØØØØØØØ)
      A(6)= ( 11796.ØØØØØØØ,   15808.ØØØØØØØ)
```

Scanned with OKEN Scanner

```
A(7)= ( 29184.0000000,  -21840.0000000)
A(8)= (-4032Ø.0000000,       Ø.0000000)

    RADACINILE APROXIMATIVE ALE ECUATIEI

Z(1)= (     Ø.9927220,       Ø.0000000)
Z(2)= (     Ø.0049455,       1.9998536)
Z(3)= (    -Ø.0067976,       5.9965420)
Z(4)= (     Ø.0067977,       7.9961410)
Z(5)= (    -7.0106702,       Ø.0029347)
Z(6)= (    -2.9653745,      -Ø.0287132)
Z(7)= (     4.9807520,       Ø.0131130)
Z(8)= (     Ø.1122120,      -3.9798708)

    RADACINILE CALCULATE CU METODA EHRLICH

    NUMAR ITERATII: 3

Z(1)= (     1.0000000,       Ø.0000000)
Z(2)= (     Ø.0000000,       2.0000000)
Z(3)= (    -Ø.0000004,       5.9999995)
Z(4)= (     Ø.0000002,       8.0000000)
Z(5)= (    -7.0000000,       Ø.0000000)
Z(6)= (    -3.0000000,       Ø.0000000)
Z(7)= (     5.0000000,       Ø.0000000)
Z(8)= (     Ø.0000000,      -4.0000000)
```

```
      PROGRAM NEWTON
      COMPLEX A,Z,Q,P,P1
      DIMENSION Z(25)
      COMMON/BT1/A(Ø:25)/BT3/N
      CALL ASSIGN(1,'SY:N.DAT')
      CALL ASSIGN(2,'SY:N.LST')
      READ(1,1Ø1) N,E
101   FORMAT(I5,F7.6)
      READ(1,1Ø2)
```

Scanned with OKEN Scanner

```
102   FORMAT(6F)
      READ(1,102) (Z(I),I=1,N)
      WRITE(2,103) E,(I,A(I),I=0,N)
103   FORMAT(1X,'REZOLVAREA ECUATIEI P(Z)=0 CU METODA NEWTON'/
      15X,'PRECIZIA IMPUSA:',F/
      25X,=COEFICIENTII POLINOMULUI'/
      3<N+1>(1X,'A(',I2,')=(',F,',',F,')'/)
      41X,'RADACINILE APROXIMATIVE ALE ECUATIEI')
      WRITE(2,104) (I,Z(I),I=1,N)
      WRITE(2,105)
105   FORMAT(1X,'RADACINILE CALCULATE CU METODA NEWTON')
      K=0
1     E1=0.
      K=K+1
      DO 2 I=1,N
      Q=P(Z(I))/P1(Z(I))
      E2=CABS(Q)
      IF(E2.GT.E1) E1=E2
2     Z(I)=Z(I)-Q
      WRITE(2,104) (I,Z(I),I=1,N)
      IF(E1.GE.E) GOTO 1
      WRITE(2,106) K
106   FORMAT(1X,'NUMAR ITERATII : ',I4)
      WRITE(2,104) (I,Z(I),I=1,N)
104   FORMAT(<N>(1X,'Z(',I2,')=(',F15.7,',',F15.7,')'/))
      STOP
      END
      COMPLEX FUNCTION P(Z)
      COMPLEX A,Z
      COMMON/ET1/A(0:25)/ET3/N
      P=A(0)
```

Scanned with OKEN Scanner

```
      DO 1 I=1,N
1     P=P*Z+A(I)
      RETURN
      END
      COMPLEX FUNCTION P1(Z)
      COMPLEX A,Z
      COMMON/ET1/A(0:25)/ET3/N
      P1=N*A(0)
      DO 1 I=1,N-1
1     P1=P1*Z+(N-I)*A(I)
      RETURN
      END
```

REZOLVAREA ECUATIEI P(Z)=0 CU METODA NEWTON

PRECIZIA IMPUSA:0.000001

COEFICIENTII POLINOMULUI

A(0) = (1.0000000, 0.0000000)
A(1) = (4.0000000, -12.0000000)
A(2) = (-46.000000, -48.000000)
A(3) = (-124.00000, 200.00000)
A(4) = (129.00000, 80.000000)
A(5) = (-624.00000, 5812.0000)
A(6) = (11796.0000, 15808.0000)
A(7) = (29184.0000, -21848.000)
A(8) = (-40320.0000, 0.0000000)

RADACINILE APROXIMATIVE ALE ECUATIEI

Z(1) = (0.9927220, 0.0000000)
Z(2) = (0.0049455, 1.9998536)
Z(3) = (-0.0067976, 5.9965420)
Z(4) = (-0.0067977, 7.9961410)
Z(5) = (-7.0106702, 0.0029347)

Scanned with OKEN Scanner

```
Z(6) = (    -2.9653745,     -Ø.Ø287152)
Z(7) = (     4.98Ø752Ø,      Ø.Ø13113Ø)
Z(8) = (     Ø.1122120,     -3.9798708)

   RADACINILE CALCULATE CU METODA NEWTON
Z(1) = (     1.ØØØØ225,      Ø.ØØØØ24Ø)
Z(2) = (    -Ø.ØØØØØØ8,      1.9999899)
Z(3) = (     Ø.ØØØØ164,      5.9999909)
Z(4) = (     Ø.ØØØØ6Ø9,      7.9999652)
Z(5) = (    -7.ØØØØ8Ø6,      Ø.ØØØØ641)
Z(6) = (    -2.9997Ø1Ø,      Ø.ØØ12664)
Z(7) = (     5.ØØØ2584,     -Ø.ØØØ43ØØ)
Z(8) = (    -Ø.ØØ22174,     -3.9883311)
.
.
.
NUMAR ITERATII : 4
Z(1) = (     1.ØØØØØØØ,      Ø.ØØØØØØØ)
Z(2) = (     Ø.ØØØØØØØ,      2.ØØØØØØØ)
Z(3) = (    -Ø.ØØØØØØ1,      6.ØØØØØØØ)
Z(4) = (     Ø.ØØØØØØ1,      7.9999995)
Z(5) = (    -7.ØØØØØØØ,      Ø.ØØØØØØØ)
Z(6) = (    -3.ØØØØØØØ,      Ø.ØØØØØØØ)
Z(7) = (     5.ØØØØØØØ,      Ø.ØØØØØØØ)
Z(8) = (     Ø.ØØØØØØØ,     -4.ØØØØØØØ)
```

```
      PROGRAM CEBIS
      COMPLEX A,Z,Q,P,P1,P2
      DIMENSION Z(25)
      COMMON/ET1/A(Ø:25)/ET3/N
      CALL ASSIGN(1,'SY:C.DAT')
      CALL ASSIGN(2,'SY:C.LST')
      READ(1,1Ø1) N,E
1Ø1   FORMAT(I5,F7.6)
```

Scanned with OKEN Scanner

```
      READ(1,102) (A(I),I=0,N)
102   FORMAT(6F)
      READ(1,102) (Z(I),I=1,N)
      WRITE(2,103) E,(I,A(I),I=0,N)
103   FORMAT(1X,'REZOLVAREA ECUATIEI P(Z)=0 CU METODA CEBISEV'/
     15X,'PRECIZIA IMPUSA:',F/
     25X,'COEFICIENTII POLINOMULUI'/
     3<N+1>(1X,'A(',I2,')=(',F,',',F,')'/)
     41X,'RADACINILE APROXIMATIVE ALE ECUATIEI')
      WRITE(2,104) (I,Z(I),I=1,N)
      WRITE(2,105)
105   FORMAT(1X,'RADACINILE CALCULATE CU METODA CEBISEV')
      K=0
1     E1=0.
      K=K+1
      DO 2 I=1,N
      Q=P(Z(I))/P1(Z(I))
      Q=Q+Q*Q*P2(Z(I))/((2.,0.)*P1(Z(I)))
      E2=CABS(Q)
      IF(E2.GT.E1) E1=E2
2     Z(I)=Z(I)-Q
      WRITE(2,104) (I,Z(I),I=1,N)
      IF(E1.GE.E) GOTO 1
      WRITE(2,106) K
106   FORMAT(1X,'NUMAR ITERATII :',I4)
      WRITE(2,104) (I,Z(I),I=1,N)
104   FORMAT(<N>(1X,'Z(',I2,')=(',F15.7,',',F15.7,')'/))
      STOP
      END
      COMPLEX FUNCTION P(Z)
      COMPLEX A,Z
```

Scanned with OKEN Scanner

```
      COMMON/ET1/A(Ø:25)/ET3/N
      P=A(Ø)
      DO 1 I=1,N
      P=P*Z+A(I)
      RETURN
      END
      COMPLEX FUNCTION P1(Z)
      COMPLEX A,Z
      COMMON/ET1/A(Ø:25)/ET3/N
      P1=N*A(Ø)
      DO 1 I=1,N-1
1     P1=P1*Z+(N-I)*A(I)
      RETURN
      END
      COMPLEX FUNCTION P2(Z)
      COMPLEX A,Z
      COMMON/ET1/A(Ø:25)/ET3/N
      P2=N*(N-1)*A(Ø)
      DO 1 I=1,N-2
1     P2=P2*Z+(N-I)*(N-I-1)*A(I)
      RETURN
      END
```

```
      REZOLVAREA ECUATIEI P(Z)=Ø CU METODA CEBISEV
          PRECIZIA IMPUSA : Ø.ØØØØØ1
          COEFICIENTII POLINOMULUI
      A(Ø) = (       1.ØØØØØØØ,       Ø.ØØØØØØØ)
      A(1) = (       4.ØØØØØØØ,     -12.ØØØØØØØ)
      A(2) = (     -46.ØØØØØØØ,     -48.ØØØØØØØ)
      A(3) = (    -124.ØØØØØØØ,     2ØØ.ØØØØØØØ)
      A(4) = (     129.ØØØØØØØ,      8Ø.ØØØØØØØ)
```

Scanned with OKEN Scanner

A(5) = (-624.0000000, 5812.0000000)
A(6) = (11796.0000000, 15808.0000000)
A(7) = (29184.0000000, -21848.0000000)
A(8) = (-40320.0000000, 0.0000000)

RADACINILE APROXIMATIVE ALE ECUATIEI

Z(1) = (0.9927220, 0.0000000)
Z(2) = (0.0049455, 1.9998536)
Z(3) = (-0.0067976, 5.9965420)
Z(4) = (-0.0067977, 7.9961410)
Z(5) = (-7.0106702, 0,0029347)
Z(6) = (-2.9653745, -0.0287132)
Z(7) = (4.9807520, 0.0131130)
Z(8) = (0.1122120, -3.9798708)

RADACINILE CALCULATE CU METODA CEBISEV

Z(1) = (1.0000000, -0.0000002)
Z(2) = (0.0020000, 2.0000000)
Z(3) = (0.0000002, 5.0000005)
Z(4) = (-0.0000001, 8.0000010)
Z(5) = (-7.0000010, 0.0000013)
Z(6) = (-3.0000610, 0.0000161)
Z(7) = (4.9999981, 0.0000161)
Z(8) = (-0.0019043, -4.0003166)

.
.
.

NUMAR ITERATII : 3

Z(1) = (1.0000000, 0.0000000)
Z(2) = (0.0000000, 2.0000000)
Z(3) = (0.0000003, 6.0000000)
Z(4) = (-0.0000002, 8.0000000)
Z(5) = (-7.0000000, 0.0000000)
Z(6) = (-3.0000000, 0.0000000)

Scanned with OKEN Scanner

Z(7) = (5.ØØØØØØØ, Ø.ØØØØØØ1)

Z(8) = (-Ø.ØØØØØØ1, -4.ØØØØØØ1)

1.6. Metode de determinare a zerourilor multiple

Metoda lui Newton pentru zerouri multiple. Fie polinomul (1.1) şi z_0^* un zerou cu ordinul $q > 1$ de multiplicitate, adică

$$P(z_0^*) = 0 \quad P'(z_0^*) = 0 \ \ldots \ P^{(q-1)}(z_0^*) = 0 \text{ şi } P^{(q)}(z_0^*) \neq 0.$$

Metoda Newton rămîne convergentă dar ordinul de convergenţă este mai mic decît 2. Considerăm funcţia $(P(z))^{1/q}$ care are pe z_0^* ca zerou simplu. Procesul Newton devine :

$$z^{k+1} = z^k - q\, P(z^k)/P'(z^k) \qquad (1.14)$$

Acest proces are ordinul de convergenţă doi [41(1)]. Metoda (1.14) presupune cunoaşterea anticipată a ordinului de multiplicitate. Ordinul de multiplicitate se poate deduce folosind algoritmul (1.4) sau (1.5). In exemplul ce urmează s-a folosit algoritmul Lehmer-Schur (1.4) cu precizia 0.1. S-au obţinut 3 zerouri care faţă de precizie cu care au fost determinate sînt zerouri multiple de ordinul 3. Putem presupune eventual că zerourile sînt multiple cu ordinul de multiplicitate 3 (adică în (1.14) facem $q = 3$). In cele ce urmează este prezentat programul NEWMUL cu determinarea iniţială a zerourilor prin metoda (1.4).

REZOLVAREA ECUATIEI P(Z)=Ø CU METODA LEHMER-SCHUR

PRECIZIA IMPUSA : Ø.1

COEFICIENTII POLINOMULUI

A(Ø) = (1.,Ø.)

A(1) = (-5.1961522,Ø.)

A(2) = (9.,Ø.)

A(3) = (-5.1961522,Ø.)

```
      RADACINILE ECUATIEI
        Z(1)=(1.7229066,-Ø.ØØØØØØ1)
        Z(2)=(1.7366228,-Ø.ØØ79392)
        Z(3)=(1.7366228,-Ø.ØØ79393)

      PROGRAM NEWMUL
      COMPLEX A,Z,X,P,P1
      COMMON/ET1/A(Ø:25)/ET3/N
      CALL ASSIGN(1,'SY:NM.DAT')
      CALL ASSIGN(2,'SY:NM.LST')
      READ(1,1Ø1) N,E,Q
1Ø1   FORMAT(I5,2F)
      READ(1,1Ø2) (A(I),I=Ø,N)
1Ø2   FORMAT(6F)
      READ(1,1Ø7) Z
1Ø7   FORMAT(2F)
      WRITE(2,1Ø3) E,(I,A(I),I=Ø,N)
1Ø3   FORMAT(1X,'REZOLVAREA ECUATIEI P(Z)=Ø CU METODA NEWTON'/
      15X,'PRECIZIA IMPUSA:',F/
      25X,'COEFICIENTII POLINOMULUI'/
     3<N+1>(1X,'A(',I2,')=(',F,',',F,')'/),
      41X,'RADACINA MULTIPLA A ECUATIEI')
      WRITE(2,1Ø4)Z
      WRITE(2,1Ø5)
1Ø5   FORMAT(1X,'RADACINA CALCULATA CU METODA NEWTON')
      K=Ø
1     K=K+1
      X=P(Z)/P1(Z)
      X=X*Q
      E1=CABS(X)
      Z=Z-X
```

Scanned with OKEN Scanner

```
      IF(E1.GE.E) GOTO 1
      WRITE(2,106) K
106   FORMAT(1X,'NUMAR ITERATII : ',I4)
      WRITE(2,104) Z
104   FORMAT(1X,'Z = (',F15.7,',',F15.7,')')
      STOP
      END
      COMPLEX FUNCTION P(Z)
      COMPLEX A,Z
      COMMON/ET1/A(0:25)/ET3/N
      P=A(0)
      DO 1 I=1,N
1     P=P*Z+A(I)
      RETURN
      END
      COMPLEX FUNCTION P1(Z)
      COMPLEX A,Z
      COMMON/ET1/A(0:25)/ET3/N
      P1=N*A(0)
      DO 1 I=1,N-1
1     P1=P1*Z+(N-I)*A(I)
      RETURN
      END
```

```
REZOLVAREA ECUATIEI P(Z)=0 CU METODA NEWTON
    PRECIZIA IMPUSA : 0.00001
    COEFICIENTII POLINOMULUI
A(0)=(      1.0000000,      0.0000000)
A(1)=(     -5.1961522,      0.0000000)
A(2)=(      9.0000000,      0.0000000)
A(3)=(     -5.1961522,      0.0000000)
```

Scanned with OKEN Scanner

RADACINA MULTIPLA A ECUATIEI

Z=(1.7229000, 0.0000000)

RADACINA CALCULATA CU METODA NEWTON

NUMAR ITERATII : 10

Z=(1.7320508, 0.0000000)

Scanned with OKEN Scanner

CAPITOLUL 2.

ECUATII CU DIFERENTE CORESPUNZATOARE PROBLEMELOR LA LIMITĂ NELINIARE

În acest capitol se consideră ecuaţiile cu diferenţe corespunzătoare unor probleme la limită neliniare : probleme la limită bilocală pentru ecuaţii diferenţiale slab neliniare şi cvasiliniare şi problema Dirichlet pentru ecuaţii eliptice slab neliniare. Ecuaţiile cu diferenţe considerate reprezintă exemple tipice de ecuaţii neliniare lacunare. Vom menţiona rezultatele lui M.Lees [35(1)] şi M.Lees şi M.H.Schultz [36(1)] relative la existenţa şi unicitatea soluţiilor ecuaţiilor cu diferenţe respective precum şi unele probleme legate de convergenţa soluţiilor problemelor discrete la soluţiile problemelor continue. De asemenea în paragraful 2.3 se construieşte o teorie de tipul M.Lees relativă la rezolvarea problemei Dirichlet pentru ecuaţii cu derivate parţiale eliptice slab neliniare prin metode cu diferenţe finite standard.

2.1. Problema la limită bilocală pentru ecuaţii diferenţiale slab neliniare.

Problema la limită bilocală pentru ecuaţii diferenţiale slab neliniare precum şi metode cu diferenţe pentru rezolvarea numerică a ei au fost considerate de P.Henrici [26(1)], M.Lees [35(1)], M.Keller [29(1)], J.Douglas [19(1)], Şamanski [62(1)] şi alţii. M.Lees a dat o teoremă de existenţă şi unicitate şi a obţinut condiţii simple de convergenţă pentru metodele cu diferenţe, rezultate pe care le vom prezenta în acest paragraf.

2.1.1. Notaţii şi noţiuni introductive.

Fie n un întreg arbitrar şi $h = (n+1)^{-1}$. Mulţimea discretă $\overline{d} = \{t_j = jh \mid j = \overline{0,n+1}\}$ o vom numi diviziunea uniformă închisă a intervalului $I = [0,1]$; $d = \overline{d} \setminus \{0,1\}$ va fi diviziunea uniformă deschisă a aceluiaşi interval. Vom nota cu X_n mulţimea funcţiilor reale x definite pe $\overline{d}$ astfel încît $x(0) = x(1) = 0$. Este clar că mulţimea X_n înzestrată cu operaţiile de adunare şi înmulţire cu scalari obişnuite, este un spaţiu liniar de dimensiune n. Dacă definim pe X_n norma $\|x\|_d = \max_{t \in \overline{d}} |x(t)|$ (numită norma uniformă sau norma max) atunci X_n devine un spaţiu Banach real n-dimensional. Mulţimea funcţiilor reale x definite pe $\overline{d}$ fără condiţiile $x(0) = x(1) = 0$, structurată în acelaşi mod, constituie un spaţiu Banach cu $n+2$ dimensiuni ; vom nota acest spaţiu cu X_{n+2} .

Observaţia 2.1. Vom identifica spaţiul X_n cu spaţiul real n-dimensional $\mathbb{R}^n$. Dacă $x \in X_n$ atunci x va fi considerat un vector coloană avînd componentele $x_j = x(t_j)$ $j = \overline{1,n}$. Vom utiliza, după caz, terminologia unui spaţiu de funcţii sau a spaţiului $\mathbb{R}^n$. Dacă $u \in C(I)$ şi $u(0) = u(1) = 0$, restricţia lui u la $\overline{d}$ o vom nota de asemenea cu u ; această dublă semnificaţie a lui u nu va provoca în general ambiguităţi ; de exemplu, prin $\|u\|_d$ vom înţelege norma uniformă a restricţiei lui u la $\overline{d}$. Menţionăm că norma (uniformă) a spaţiului $C(I)$ o vom nota cu $\|\cdot\|_I$.

Fie $x, y \in X_n$, vom nota, pentru orice k , $0 \leq k \leq n+1$.

$$\langle x,y \rangle_k = h \sum_{j=1}^{k} x(t_j).y(t_j),$$

Se vede că $\langle \cdot,\cdot \rangle_n$ defineşte un produs interior pe spaţiul X_n. Norma indusă de acest produs o vom nota cu $\|\cdot\|_0 = \langle x,x \rangle_n^{1/2}$. Se poate verifica cu uşurinţă că

$$\sqrt{h}\,\|x\|_d \leq \|x\|_o \leq d\|x\|_d$$

Fie $A : X_n \to X_n$ un operator definit de

$$Ax(t) = h^{-2}(x(t+h)-2x(t)+x(t-h)) \quad , \quad t \in d$$

și $Ax(o) = Ax(1) = o$. Se știe că A este un operator liniar cu matricea (notată de asemenea cu A)

$$A = h^{-2}\begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & \cdot & & \cdot \\ 0 & 0 & 0 & 0 & \dots & -2 \end{bmatrix} \qquad (2.1)$$

Să observăm că, dacă $u \in C^1(I)$, Au constituie diferența divizată de ordinul doi a funcției u în raport cu pasul h. După cum se știe această diferență aproximează, în anumite condiții derivata de ordinul doi a funcției u în punctele diviziunii d ; acest fapt este esențial pentru aproximarea soluțiilor problemelor la limită bilocale (liniare sau neliniare).

Se definesc de asemenea operatorii diferență D_+ și D_- pe spațiul X_{n+2} cu valori în X_{n+2} cu ajutorul relațiilor

$$D_+x(t) = h^{-1}(x(t+h)-x(t))$$

$$D_-x(t) = h^{-1}(x(t)-x(t-h))$$

pentru orice $t \in d$ și $D_+x(1) = -x(1)$, $D_-x(o) = x(o)$. Matricile acestor operatori sînt :

$$D_+ = h^{-1}\begin{bmatrix} -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ 0 & 0 & -1 & \dots & 0 \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & 1 \\ 0 & 0 & 0 & \dots & -1 \end{bmatrix} \qquad D_- = h^{-1}\begin{bmatrix} 1 & 0 & 0 & \dots & 0 \\ -1 & 1 & 0 & \dots & 0 \\ 0 & -1 & 1 & \dots & 0 \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ \cdot & \cdot & \cdot & & \cdot \\ 0 & 0 & 0 & -1 & 1 \end{bmatrix}$$

D_+x și D_-x reprezintă diferențele divizate de ordinul întîi

("la dreapta" şi respectiv "la stînga") ale funcţiei x în raport cu pasul h. Dacă notăm cu $A_M = D_+D_-$ atunci $A_M(X_n) \subset X_n$ şi restricţia lui A_M la X_n coincide cu A. Operatorul $-A$ fiind simetric şi pozitiv definit, se poate defini pe spaţiul X_n norma $\|\cdot\|_A$ dată de

$$\|x\|_A^2 = - \langle Ax, x \rangle_n$$

Este evident că $\|x\|_A^2 = - \langle A_M x, x \rangle_n = - \langle D_+D_-x, x \rangle_{n+1} = \langle D_-x, D_-x \rangle_{n+1}$. Utilizînd această relaţie se poate arăta (M.Lees [35(1)], lema 2) că pentru orice $x \in X_n$ avem :

$$2\|x\|_d \leq \|x\|_A$$

2.1.2. Problema la limită bilocală slab neliniară.

Vom considera următoarea problemă la limită bilocală slab neliniară

$$\begin{cases} u'' = f(t,u) \\ u(0) = a \ , \ u(1) = b \end{cases} \qquad (2.2)$$

$f : S \longrightarrow \mathbb{R}$, $S = [0,1] \times \mathbb{R}$, $a,b \in \mathbb{R}$. Să observăm că limitele a,b se pot considera întotdeauna egale cu $0 \in \mathbb{R}$; într-adevăr, dacă u este o soluţie a problemei (2.2) şi notăm cu $g(t,v) = f(t,v + \xi)$, $\xi(t) = bt + (1-t)a$ atunci $v = u - \xi$ este o soluţie a problemei

$$\begin{cases} v'' = g(t,v) \\ v(0) = v(1) = 0 \end{cases}$$

Invers orice soluţie v a acestei probleme generează o soluţie $u = v + \xi$ a problemei (2.2). In cele ce urmează considerăm problema (2.2) cu $a = b = 0 \in \mathbb{R}$. Următoarea teoremă dată de către M.Lees [35(1)] constituie unul din rezultatele principale asupra existenţei şi unicităţii soluţiei problemei (2.2) :

Teorema 2.1. Presupunem că f are derivate parţiale de orice ordin în raport cu u continue şi că

$$\inf_S f_u = -\eta > -\pi^2 \qquad (2.3)$$

Atunci problema la limită bilocală (2.2) are o soluţie unică.. Această teoremă generalizează un rezultat prezentat de P.Henrici [26(1)] care presupune că f_u este continuă şi satisface relaţia $0 \leq f_u(t,u) \leq L$ pe S pentru o anumită constantă L. De remarcat faptul că condiţia (2.3) presupune că f_u este mărginită numai la stînga.

2.1.3. Ecuaţii cu diferenţe finite corespunzătoare problemei (2.2). Să presupunem că problema (2.2) admite o soluţie $u \in C^4(I)$. Aplicînd formula lui Taylor se obţine

$$Au(t) = u''(t) + \frac{h^2}{12} u^{(4)}(\xi) \qquad 0 < \xi < 1$$

Ţinînd cont că $u^{(4)}$ este mărginită pe I, avem $u'' \approx Au$, astfel încît în locul ecuaţiei $u'' = f(t,u)$ vom considera ecuaţia $Au = f(t,u)$. In mod obişnuit soluţia acestei ecuaţii este apropiată de soluţia ecuaţiei iniţiale, dar rezolvarea ei nu este în general mai simplă. De aceea vom căuta o soluţie x a acestei ecuaţii în spaţiul X_n şi vom numi problema astfel obţinută problema discretă corespunzătoare problemei continue (2.2). Dacă notăm cu $\Phi x = [f(t_1,x(t_1)),\ldots,f(t_n,x(t_n))]^T$ ecuaţia discretă se poate scrie sub forma :

$$Ax - \Phi x = 0 \qquad (2.4)$$

unde A are forma (2.1). Este clar că (2.4) constituie un sistem nelinier de n ecuaţii cu n necunoscute. Vom numi acest sistem : ecuaţii cu diferenţe finite standard corespunzătoare problemei la limită bilocale (2.2).

Asupra ecuaţiilor (2.4) se pun următoarele trei probleme:

1. Existenţa şi unicitatea soluţiei sistemului
2. Convergenţa soluţiilor sistemului de forma (2.4) într-un anumit sens, la soluţia proolemei (2.2) atunci cînd $n \to \infty$.
3. Rezolvarea numerică a sistemului (2.4).

Scanned with OKEN Scanner

Menţionăm că, avînd în vedere forma particulară a ecuaţiilor (2.4), se pot construi proceduri speciale de calcul în vederea utilizării unui calculator electronic care să asigure un consum redus de memorie şi de timp unitate centrală. Această problemă va fi tratată în capitolele 3 şi 4.

Să presupunem acum că $u \in C^6(I)$ şi să considerăm dezvoltarea în serie Taylor a funcţiei u termenii pînă la ordinul 6 inclusiv. Se obţine

$$Au(t) = u''(t) + \frac{h^2}{12} u^{(4)}(t) + \frac{h^4}{360} u^{(6)}(\xi_1), \quad 0 < \xi_1 < 1$$

Ţinînd cont că :

$$u^{(4)}(t) = \frac{d^2}{dt^2} f(t,u(t)) = Af(t,u(t)) - \frac{h^2}{12} \frac{d^4}{dt^4} f(t,u(t))_{t=\xi_2}, \quad 0 < \xi_2 < 1$$

rezultă că

$$Au(t) = u''(t) + \frac{h^2}{12} Af(t,u(t)) + \frac{h^4}{12} \left(\frac{1}{30} u^{(6)}(\xi_1) - \frac{d^4}{dt^4} f(t,u(t))_{t=\xi_2}\right).$$

Aşadar $Au \approx f(t,u)+(h^2/12)Af(t,u)$ şi rezultă imediat următoarea discretizare pentru (2.2)

$$Ax - (I - qA)\,\Phi x = o \qquad (2.5)$$

unde $q = \frac{h^2}{12}$. S-a obţinut din nou un sistem liniar de n ecuaţii cu n necunoscute care diferă de (2.4) prin partea neliniară. Vom numi acest sistem ecuaţiile cu diferenţe finite corectate corespunzătoare problemei la limită bilocală (2.2).

Dacă presupunem că u soluţia problemei (2.2) este de clasă $C^8(I)$ atunci în mod analog se pot construi ecuaţiile cu diferenţe finite cu corecţii de ordinul 2 corespunzătoare problemei la limită bilocală (2.2)

$$Ax - (I - qA + rA^2)\,\Phi x = o \qquad (2.6)$$

unde $r = h^4/240$.

Ordinul de aproximare este $O(h^6)$.

2.1.4. Convergenţa. Vom nota cu $C_0(I)$ mulţimea funcţiilor reale continue pe I şi astfel încît $u(0) = u(1) = 0$. Dacă

$u \in C_o(I)$ şi $x \in X_n$ vom înţelege prin $u-x$ diferenţa dintre restricţia lui u la d şi x. Fie acum $\{x^n\}$ un şir astfel încît $x^n \in X_n$. Vom spune că şirul $\{x^n\}$ converge la u $C_o(I)$ dacă $\|x^n - u\|_d \to o$ cînd $n \to \infty$.

Teoria lui Melton Lees referitoare la rezolvarea numerică a problemei (2.2) prin metoda cu diferenţe are la bază următoarea lemă (lema 6, [35,(1)]) :

Lema 2.1. Presupunem că f satisface relaţia (2.3) şi că h_o este astfel încît

$$\eta < \pi^2 \left[1 - h_o^2 \pi^2/12\right] .$$

Dacă $h \leq h_o$ şi $x, y \in X_n$ satisface relaţiile

$$Ax = f(t,x) + q_1(t)$$
$$Ay = f(t,y) + q_2(t)$$

pentru $t \in d$ atunci $\|x-y\|_A \leq 2k_o \|q_1 - q_2\|_o$, unde

$$2k_o = \frac{\pi}{\pi^2 \left[1 - h_o^2 . \pi^2/12\right] - \eta}$$

Pe baza acestei leme se poate obţine următorul rezultat ([35(1)], teoremele 2 şi 3) asupra convergenţei soluţiilor ecuaţiilor cu diferenţe finite standard la soluţia problemei (2.2).

Teorema 2.2. Presupunem că f satisface relaţia (2.3) şi că $h \leq h_o$. Atunci ecuaţiile cu diferenţe finite standard (2.4) au o soluţie unică $x^n \in X_n$. Dacă u este soluţia problemei (2.2) atunci

$$\|u - x^n\|_d \leq \frac{1}{2} \|u - x^n\|_d \leq \frac{h^2}{12} k_o \|u^{(4)}\|_I .$$

Să observăm că din această teoremă rezultă că dacă $n \to \infty$ atunci $h \to o$ şi $\|u - x^n\|_d \to o$, adică şirul de soluţii ale ecuaţiilor cu diferenţe finite standard converge la soluţia problemei (2.2).

Relativ la ecuaţiile cu diferenţe finite cu corecţii de ordinul 1 se obţine ([35(1)] teoremele 5 şi 6).

Teorema 2.3. Dacă f satisface relaţia (2.3) atunci ecuaţiile cu diferenţe finite cu corecţii de ordinul 1 (2.5) au o soluţie unică $x^n \in X_n$ pentru h suficient de mic iar şirul de soluţii ale acestor ecuaţii converge la soluţia problemei (2.2).

In mod natural are loc şi teorema :

Teorema 2.4. Dacă f satisface relaţia (2.3) atunci ecuaţiile cu diferenţe finite cu corecţii de ordinul 2 (2.6) au o soluţie unică $x^n \in X_n$ pentru h suficient de mic iar şirul de soluţii ale acestor ecuaţii converge la soluţia problemei (2.2).

2.2. Problema la limită bilocală pentru ecuaţii diferenţiale cvasiliniare.

In acest paragraf vom prezenta rezultatele lui Melton Lees şi M.H.Schultz [36(1)] relative la rezolvarea numerică a problemei la limită bilocală pentru ecuaţii diferenţiale cvasiliniare prin metode cu diferenţe finite. Aceste rezultate au fost obţinute pe baza generalizării principiului de punct fix Leray-Schauder pentru operatori A-compacţi, teorie dezvoltată de W.V.Petryshyn [57(2)], S.Kaniel [27(1)], M.Lees şi M.H.Schultz [36(1)].

2.2.1. Noţiuni introductive. Vom nota cu $C_0^1(I)$ spaţiul liniar al funcţiilor reale u, continuu diferenţiabile pe $I = [0,1]$ şi astfel încît $u(0) = u(1) = 0$. Dacă pe $C_0^1(I)$ definim norma

$$\|u\| = \max_I |u(t)| + \max_I |u'(t)|$$

atunci $C_0^1(I)$ devine un spaţiu Banach real. Pe spaţiul X_n vom considera norma

$$\|x\|_d = \max_d |x(t)| + \max_d |D_- x(t)|$$

unde D_- este operatorul diferenţă definit în paragraful 2.1.1. Spaţiul X_n înzestrat cu această normă devine de asemenea un spaţiu Banach real n-dimensional.

Pentru fiecare X_n definim un operator $\varphi_n : X_n \to C_o^1(I)$, $(x \to u = \varphi_n(x))$ după cum urmează :

1) dacă $t \in [t_j, t_{j+1}]$, $1 \leq j \leq n$, atunci $u = \varphi_n(x)$ este polinomul (unic) de grad cel mult trei astfel încît

$$u(t_j) = x(t_j) \quad , \quad u'(t_j) = D_-x(t_j)$$

$$u(t_{j+1}) = x(t_{j+1}) \quad , \quad u'(t_{j+1}) = D_-x(t_{j+1}) \; ;$$

2) dacă $t \in [o, t_1]$ atunci $u = \varphi_n(x)$ este polinomul (unic) de grad cel mult doi astfel încît

$$u(o) = o$$

$$u(t_1) = x(t_1) \quad u'(t_1) = D_- x(t_1).$$

Să observăm că pe fiecare interval $[t_j, t_{j+1}]$, $1 \leq j \leq n$ (respectiv pe $[O, t_1]$), funcţia $u \in C_o^1(I)$, care corespunde elementului $x \in X_n$ este definită de un polinom de interpolare Hermite de ordinul trei (respectiv de ordinul doi).

Se poate arăta că φ_n este un operator liniar şi că există o constantă c_1, independentă de n, astfel încît :

$$\|x\|_d \leq \|\varphi_n x\| \leq c_1 \|x\|_d \quad \forall\, x \in X_n \qquad (2.7)$$

2.2.2. Problema la limită bilocală cvasiliniară şi ecuaţii cu diferenţe finite standard. Considerăm următoarea problemă la limită bilocală cvasiliniară

$$\begin{cases} u'' = f(t,u,u') \quad , \quad t \in [o,1] \\ u(o) = u(1) = o \end{cases} \qquad (2.8)$$

unde f este o funcţie reală definită pe $S = \{(t,u,u') \mid o \leq t \leq 1, \; -\infty < u, u' < +\infty\}$. Dacă presupunem că problema (2.8) are o soluţie $u \in C^4(I)$ iar f are derivată parţială în raport cu u' mărginită, atunci

$$Au(t) = f(t, u(t), D_o u(t)) + r \qquad t \in d$$

unde $D_o u(t) = (u(t+h) - u(t-h))/2h$ iar r satisface $|r| \leq c_2 h^2$ pentru o constantă c_2 independentă de n. Aşadar $Au \cong$

$\cong f(t,u,D_0u)$ şi dacă notăm

$$\Phi(x) = \begin{bmatrix} f(t_1,x_1,(x_2-x_0)/2h) \\ f(t_2,x_2,(x_3-x_1)/2h) \\ \cdot \\ \cdot \\ \cdot \\ f(t_n,x_n,(x_{n+1}-x_{n-1})/2h) \end{bmatrix} \qquad (2.9)$$

rezultă discretizarea (2.4) pentru problema (2.8). Ca şi în cazul slab neliniar, ecuaţiile (2.4) cu Φ dat de (2.9) le vom numi ecuaţiile cu diferenţe standard corespunzătoare problemei la limită bilocală cvasiliniară (2.8).

Vom presupune că f este de două ori continuu diferenţiabilă pe S ; orice soluţie a proolemei (2.8) este în acest caz de clasă C^4 pe $[o,1]$. Asupra funcţiei f se mai impun următoarele condiţii :

Dacă u este o soluţie a problemei (2.8) atunci :

$$f_x(t,u,o) \geq m > o$$
$$|f(t,u,u')| \leq \sigma(u') \qquad (2.1o)$$

pentru $|u| \leq k_1$ unde $k_1 = m^{-1}.\max|f(t,o,o)|$ iar σ este o funcţie pozitivă şi continuă pe $(o,+\infty)$ astfel încît

$$\int_o^\infty \frac{sds}{\sigma(s)} = +\infty .$$

Se poate arăta că în aceste condiţii $|u(t)| \leq k_1$ şi $|u'(t)| \leq k_2$ unde k_2 este o constantă care depinde numai de k_1 .

Putem acum enunţa următoarea teoremă asupra problemei (2.8) şi a metodei cu diferenţe finite pentru rezolvarea numerică a ei (M.Lees şi M.H.Schultz [76,(1)] , teorema 7) :

Teorema 2.5. Dacă f este o funcţie de două ori continuu diferenţiabilă pe S şi satisface condiţiile (2.1o) atunci problema la limită (2.8) are cel puţin o soluţie u, ecuaţiile cu diferenţe standard corespunzătoare au o soluţie x^n pentru n sufi-

cient de mare şi există un subşir $\{x^{n_j}\}$ al lui $\{x^n\}$ astfel încît $\|\varphi_{n_j} x^{n_j} - u\| \to o$, $n_j \to \infty$.

Dacă în plus $f_x(t,x,x') > o$ atunci se poate arăta că soluţia u a problemei (2.8) este unică şi că şirul $\{\varphi_n x^n\}$ converge la u în topologia spaţiului $C_o^1(I)$. Aproximarea obţinută cu ajutorul soluţiei x^n a problemei discrete, pe mulţimea d, este de ordinul $o(h^2)$.

Se poate obţine de asemenea un rezultat similar cu cel din teorema (2.2) pentru cazul slab neliniar. Deoarece şirul polinoamelor de interpolare Hermite corespunzătoare unei funcţii, respectiv derivatele lor, converge uniform la funcţia considerată, respectiv la derivata ei, se poate arăta că $\|\varphi_n u - u\| \to o$, $n \to \infty$. Ţinînd cont şi de relaţia (2.7) rezultă

$$\|x^n - u\|_d \leq \|\varphi_n(x^n - u)\| \leq \|\varphi_n x^n - u\| + \|u - \varphi_n u\| \to o$$

ceea ce înseamnă că şirul de soluţii x^n al ecuaţiilor cu diferenţe finite converge la soluţia problemei (2.8).

2.2.3. Caz particular. Să considerăm problema cvasiliniară particulară de forma :

$$\begin{cases} u'' = a(u)u + f(t,u) \\ u(o) = u(1) = o \end{cases} \qquad (2.11)$$

unde a şi f sînt funcţii reale definite respectiv pe $\mathbb{R}$ şi $[o,1] \times \mathbb{R}$. Dacă notăm $a(x) = \mathrm{Diag}[a(x_1),\ldots,a(x_n)]$ ecuaţiile cu diferenţe standard se pot scrie sub forma

$$[A - a(x)]x - \Phi x = o$$

unde A şi Φ au semnificaţiile din paragraful 2.1.1. Dacă notăm $A(x) = A - a(x)$, aceste ecuaţii devin :

$$A(x).x - \Phi x = o \qquad (2.12)$$

Sub această formă ecuaţiile cu diferenţe prezintă anumite avantaje din punctul de vedere al algoritmului de calcul numeric (paragrafele 3.4, 3.5, 3.6).

2.3. Problema Dirichlet pentru ecuaţii eliptice slab neliniare.

In acest paragraf vom construi o teorie de tipul M.Lees relativă la rezolvarea problemei Dirichlet pentru ecuaţii cu derivate parţiale eliptice slab neliniare prin metode cu diferenţe finite standard (în cinci puncte). Această problemă a fost considerată de L.Bers [7(1)], N.Levinson [38(1)], S.V.Parter [53(1),(2)], J.Douglas [19(1)]. L.Bers, într-o lucrare cunoscută [7(1)] a arătat că soluţiile ecuaţiilor cu diferenţe finite standard converg la soluţia problemei continue în condiţii suficient de largi, principala condiţie fiind (2.14); aceste rezultate vor fi generalizate [58(5)] în capitolul de faţă. Intr-o serie de lucrări [53(1),(2)] , S.V.Parter a studiat metodele cu diferenţe finite standard şi cu diferenţe finite corectate pentru obţinerea unei soluţii "maximale" (în cazul în care problema admite mai multe soluţii). Menţionăm de asemenea rezultatele lui J.Douglas [19(1)] asupra metodei direcţiilor alternate pentru ecuaţiile cu diferenţe finite standard ; condiţia principală impusă de J.Douglas condiţia (2.15), mai slabă decît condiţia cerută de L.Bers, va fi utilizată şi în studiul nostru ca şi condiţie principală.

2.3.1. Notaţii şi noţiuni introductive

Fie $D = (0,1)\times(0,1)$ pătratul unitate din $\mathbb{R}^2$, Γ frontiera lui D şi $\overline{D} = D \cup \Gamma$. Vom nota cu $C_0^n(\overline{D})$ mulţimea funcţiilor reale u, de n ori continuu diferenţiabile pe D şi astfel încît $u|_\Gamma = 0$. Pentru simplificarea scrierii, un punct oarecare $(t;s) \in D$ va fi notat P. Vom considera următoarea problemă Dirichlet pentru ecuaţii cu derivate parţiale, eliptice slab neliniare (pe scurt problema Dirichlet eliptică slab neliniară),

$$\begin{cases} \Delta u = f(P,u) & , \quad P \in D \\ u|_\Gamma = 0 \end{cases} \tag{2.13}$$

unde $\Delta u = \frac{\partial^2 u}{\partial t^2} + \frac{\partial^2 u}{\partial s^2}$, iar $f : D \times \mathbb{R} \to \mathbb{R}$. Dacă f satisface anumite condiţii de "netezire" (de exemplu [53(1)]) iar f_u este pozitivă

$$f_u(P,u) \geq 0 \qquad (P,u) \in D \times \mathbb{R} \qquad (2.14)$$

atunci problema (2.13) are soluţie unică. De asemenea ecuaţiile cu diferenţe finite standard au soluţii unice iar şirul soluţiilor acestor ecuaţii converge la soluţia problemei (2.13). J.Douglas [19(1)] a studiat metoda direcţiilor alternante pentru rezolvarea numerică a ecuaţiilor cu diferenţe finite, impunînd condiţia mai slabă

$$f_u(P,u) \geq -\eta > -2\pi^2 \quad , \quad (P,u) \in D \times \mathbb{R} \qquad (2.15)$$

Fie m un întreg arbitrar şi $h = (m+1)^{-1}$. Ca şi în cazul unidimensional, mulţimea discretă $\bar{d} = \{(t_i,s_j) | t_i = ih \ , \ s_j = jh \ , \ i, j = \overline{0,m+1}\}$ o vom numi reţeaua uniformă închisă a pătratului unitate $\bar{D}$; $d = \{(t_i,s_j) | t_i = ih, \ s_j = jh \ , \ i,j = \overline{1,m}\}$ va fi reţeaua uniformă deschisă a lui $\bar{D}$. Vom nota $\delta = \bar{d} - d$ şi vom spune că δ este frontiera lui $\bar{d}$. Este clar că d conţine $n = m^2$ puncte din pătratul unitate $\bar{D}$ iar $\bar{d}$, $n_1 = (m+2)^2$ puncte din acelaşi pătrat. Vom nota cu X_n mulţimea funcţiilor reale x definite pe $\bar{d}$ astfel încît $x(P) = 0$ pentru $P \in \delta$ şi cu X_{n_1} funcţiile x fără această condiţie. Structurate ca şi în paragraful 2.1, X_n şi X_{n_1} vor fi spaţii Banach reale finit dimensionale (cu n şi respectiv n_1 dimensiuni). Ca şi în cazul unidimensional, vom identifica spaţiul X_n cu spaţiul real n-dimensional $\mathbb{R}^n$. Un element x din X_n considerat ca vector coloană din $\mathbb{R}^n$ va avea componentele $x_{ij} = x(t_i,s_j)$ $i,j = \overline{1,m}$ unde convenim ca primul indice care se modifică să fie i, adică $x = (x_{11},x_{21},\ldots,x_{m1},\ldots,x_{1m},\ldots,x_{mm})^T$. Aceeaşi observaţie este valabilă şi pentru X_{n_1}. Dacă $u \in C_0(\bar{D})$ restricţia lui u la $\bar{d}$ o vom nota de asemenea cu u.

Scanned with OKEN Scanner

Fie $x,y \in X_{n_1}$; vom nota, pentru $k \geq 0$, $\ell \leq m+1$, $k \leq \ell$.

$$\langle x,y \rangle_{k,\ell} = \sum_{i,j=k}^{\ell} x(t_i,s_j)\, y(t_i,s_j).$$

Se vede că $\langle \cdot,\cdot \rangle_{1,m}$ defineşte un produs interior pe spaţiul X_n. Norma indusă de acest produs o vom nota cu $\| \cdot \|_o$. Se poate verifica cu uşurinţă că

$$\sqrt{h}\,\|x\|_d \leq \|x\|_o \leq \frac{1}{\sqrt{h}}\,\|x\|_d \qquad (2.16)$$

2.3.2. Diferenţe finite.

Fie $A : X_n \longrightarrow X_n$ operatorul definit de

$Ax(t,s)=h^{-2}[x(t+h,s)+x(t,s+h)-4x(t,s)+x(t-h,s)+x(t,s-h)]$ $(t,s) \in d$

$Ax(t,s)= o \quad (t,s) \in \delta$

Se poate verifica, că A este liniar şi că matricea lui A, de dimensiuni $n \times n$, $(m^2 \times m^2)$, este constituită din blocuri de dimensiuni $m \times m$ şi are forma :

$$A = h^{-2}\begin{pmatrix} B & I & & 0 \\ I & \ddots & \ddots & \\ & \ddots & \ddots & I \\ 0 & & I & B \end{pmatrix}, \text{ unde } B = \begin{pmatrix} -4 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & -4 \end{pmatrix} \qquad (2.17)$$

ier I este matricea unitate $m \times m$ dimensională.

Dacă $u \in C_0^n(D)$ se poate arăta că $\Delta u \cong Au$ pe $\overline{d}$ (relaţia (2.18), relaţia pe baza căreia se obţin ecuaţiile cu diferenţe finite standard corespunzătoare problemei (2.13)).

Vom considera de asemenea operatorii diferenţă de ordinul întîi în raport cu t, D_+^t , D_-^t : $X_{n_1} \longrightarrow X_{n_1}$ şi respectiv în raport cu s, D_+^s , D_-^s : $X_{n_1} \longrightarrow X_{n_1}$.

Operatorii D_+^t , D_-^t sînt definiţi de

$D_+^t x(t,s) = h^{-1}[x(t+h,s)-x(t,s)]$ pe $\bar{d} - \{(1,s_j),\ j = 0,\dots,m+1\}$

$D_+^t x(1,s) = -h^{-1}x(1,s)$ $\quad s \in \{s_j = jh,\ j = 0,\dots,m+1\}$.

$D_-^t x(t,s) = h^{-1}[x(t,s)-x(t-h,s)]$ pe $\bar{d} - \{(0,s_j),\ j = 0,\dots,m+1\}$

$D_-^t x(0,s) = h^{-1}x(0,s)$ $\quad s \in \{s_j,\ j = 0,\dots,m+1\}$

Se poate verifica uşor că D_+^t, D_-^t sînt liniari şi că

$$D_+^t = h^{-1}\begin{bmatrix} C & & 0 \\ & \ddots & \\ 0 & & C \end{bmatrix};\quad D_-^t = h^{-1}\begin{bmatrix} -C^T & & 0 \\ & \ddots & \\ 0 & & -C^T \end{bmatrix} \text{ unde } C = \begin{bmatrix} -1 & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & & -1 \end{bmatrix}$$

dimensiunile lui C fiind $(m+2)\times(m+2)$.

Operatorii D_+^s, D_-^s se definesc în mod analog şi anume

$D_+^s x(t,s) = h^{-1}[x(t,s+h)-x(t,s)]$ pe $\bar{d} - \{(t_i,1),\ i = \overline{0,m+1}\}$

$D_+^s x(t,1) = -h^{-1}x(t,1)$ $\quad t \in \{t_i,\ i = \overline{0,m+1}\}$

$D_-^s x(t,s) = h^{-1}[x(t,s)-x(t,s-h)]$ pe $d - \{(t_i,0),\ i = \overline{0,m+1}\}$

$D_-^s x(t,0) = h^{-1}x(t,0)$ $\quad t \in \{t_i,\ i = \overline{0,m+1}\}$

unde D_+^s şi D_-^s sînt de asemenea liniari şi

$$D_+^s = h^{-1}\begin{bmatrix} -I & I & & 0 \\ & \ddots & \ddots & \\ & & \ddots & I \\ 0 & & & -I \end{bmatrix} \qquad D_-^s = h^{-1}\begin{bmatrix} I & & & 0 \\ I & \ddots & & \\ & \ddots & \ddots & \\ 0 & & -I & I \end{bmatrix}$$

unde I este matricea unitate $(m+2)\times(m+2)$ dimensională.

Vom nota $A_M = D_+^t D_-^t - D_+^s D_-^s$. Se poate arăta că A_M este constituită din blocuri de dimensiuni $(m+2)\times(m+2)$ şi că are forma

$$\tilde{A}_M = h^{-2}\begin{bmatrix} B_1 & I & & 0 \\ I & \ddots & \ddots & \\ & \ddots & B_1 & I \\ 0 & & I & B_2 \end{bmatrix} \text{ unde } B_1 = \begin{bmatrix} -4 & 1 & & 0 \\ 1 & \ddots & \ddots & \\ & \ddots & \ddots & 1 \\ 0 & & 1 & -4 \end{bmatrix} \quad B_2 = \begin{bmatrix} -3 & 1 & & 0 \\ & \ddots & \ddots & \\ & & \ddots & 1 \\ 0 & & 1 & -2 \end{bmatrix}$$

Scanned with OKEN Scanner

B_1 și B_2 având dimensiunile (m+2)x(m+2). Restricțiile lui A_M la X_n coincide cu A.

Vom introduce acum pe spațiul X_n norme $\|x\|_A^2 = -\langle Ax,x\rangle_{1,m}$ ținînd cont că $-A$ este simetrică și pozitiv definită ([49(1)], [20(1)]) se poate arăta că într-adevăr $\|\cdot\|_A$ satisface condițiile unei norme (pentru inegalitatea $\|x+y\|_A \leq \|x\|_A + \|y\|_A$ se utilizează inegalitatea lui Schwarz generalizată).

Să observăm că, deoarece $D_+^t = -(D_-^t)^T$ și $D_+^s = -(D_-^s)^T$ avem pentru orice x din X_n .

$$\|x\|_A^2 = -\langle Ax,x\rangle_{1,m} = -\langle A_M x,x\rangle_{0,m+1} = \langle D_-^t x, D_-^t x\rangle_{0,m+1} + \langle D_-^s x, D_-^s x\rangle_{0,m+1} .$$

Lema 2.2. Dacă $x \in X_n$ atunci $2\sqrt{2}\,\|x\|_d \leq \|x\|_A$.

Demonstrație. Fie k, ℓ doi întregi astfel încît $1 \leq k, \ell \leq m$.

Se poate verifica, că :

$$2x_{k\ell} = h\sum_{i=1}^{k} D_-^t x_{i\ell} - h\sum_{i=k+1}^{m+1} D_-^t x_{i\ell} = h\sum_{i=1}^{m+1} \varepsilon_i D_-^t x_{i\ell} \qquad \varepsilon_i = \begin{cases} 1 & i \leq k \\ -1 & i > k \end{cases}$$

$$2x_{k\ell} = h\sum_{j=1}^{\ell} D_-^s x_{kj} - h\sum_{j=\ell+1}^{m+1} D_-^s x_{kj} = h\sum_{j=1}^{m+1} \varepsilon'_j D_-^s x_{kj} , \quad \varepsilon'_j = \begin{cases} 1 & j \leq \ell \\ -1 & j > \ell \end{cases}$$

De aici rezultă :

$$4x_{k\ell} = h\sum_{i=1}^{m+1} \varepsilon_i D_-^t x_{i\ell} + h\sum_{j=1}^{m+1} \varepsilon'_j D_-^s x_{kj}$$

Utilizînd inegalitatea lui Schwarz și faptul că $\sum_{i=1}^{m+1} h = 1$, se obține

$$4|x_{k\ell}| \leq h\sum_{i=1}^{m+1} |D_-^t x_{i\ell}| + h\sum_{j=1}^{m+1} |D_-^s x_{kj}| = \sum_{i=1}^{m+1} h^{1/2} h^{1/2} |D_-^t x_{i\ell}| +$$

$$+ \sum_{j=1}^{m+1} h^{1/2} h^{1/2} |D_-^s x_{kj}| \leq \Big(\sum_{i=1}^{m+1} h\Big)^{1/2} \Big(h\sum_{i=1}^{m+1} |D_-^t x_{i\ell}|^2\Big)^{1/2} +$$

$$+ \Big(\sum_{j=1}^{m+1} h\Big)^{1/2} \Big(h\sum_{j=1}^{m+1} |D_-^s x_{kj}|^2\Big)^{1/2} = \Big(h\sum_{i=1}^{m+1} |D_-^t x_{i\ell}|^2\Big)^{1/2} + \Big(h\sum_{i=1}^{m+1} |D_-^s x_{kj}|^2\Big)^{1/2}$$

$$\leq \Big(h\sum_{i,j=0}^{m+1}|D_-^t x_{ij}|^2\Big)^{1/2}+\Big(h\sum_{i,j=0}^{m+1}|D_-^s x_{ij}|^2\Big)^{1/2}=\langle D_-^t x, D_-^t x\rangle^{1/2}+$$

$$+\langle D_-^s x, D_-^s x\rangle^{1/2}.$$

Ținînd cont că pentru orice a,b pozitivi avem $a^{1/2}+b^{1/2} \leq \sqrt{2}\,(a+b)^{1/2}$ rezultă că

$$4|x_{kl}| \leq \sqrt{2}\big(\langle D_-^t x, D_-^t x\rangle+\langle D_-^s x, D_-^s x\rangle\big)^{1/2} = \sqrt{2}\,\|x\|_A$$

și lema este demonstrată.

2.3.3. Ecuații cu diferențe finite standard.

Să presupunem că problema (2.13) are o soluție u de patru ori continuu diferențiabilă pe $\bar{D}$. Atunci are loc ([2o(1)], teorema 2o.8).

$$Au(P)-\Delta u(P) = \frac{h^2}{12}\big[u_{t^4}(\xi_t,s)+u_{s^4}(t,\xi_s)\big] \qquad (2.18)$$

unde $t-h<\xi_t<t+h$, $s-h<\xi_s<s+h$. De aici rezultă că $\Delta u \approx Au$, astfel încît în locul ecuației $\Delta u = f(P,u)$ din problema (2.13) vom considera ecuația $Au = f(P,u)$. Vom determina o soluție a ei în X_n, rezultînd astfel problema la limită discretă.

$$\begin{cases} Ax = f(P,x) & P\in d \\ x|_\gamma = o \end{cases} \qquad (2.19)$$

Dacă notăm, ca și în cazul unidimensional $\Phi(x) = (f(P_{11},x(P_{11})),\dots,f(P_{m1},x(P_{m1})),\dots,f(P_{1m},x(P_{1m})),\dots,f(P_{mm},x(P_{mm})))^T$ obținem următorul sistem de ecuații neliniare

$$Ax - \Phi x = o \qquad (2.2o)$$

Cu A dat de (2.17), sistem care constituie ecuațiile cu diferențe finite standard corespunzătoare problemei (2.13), cu un ordin de aproximare de $o(h^2)$.

Dacă presupunem că problema (2.13) are o soluție u de 6 ori continuu diferențiabilă pe $\bar{D}$, atunci în mod asemănător are loc

$$Au(P) - \frac{h^2}{12} Af(P,u(P)) - \Delta u(P) \leq o(h^4)$$

deci în locul ecuaţiei $\Delta u = f(P,u)$ din problema (2.13) vom considera ecuaţia $Au - qAf(P,u) = f(P,u)$, unde $q = h^2/12$. Vom determina o soluţie a ei în X_n, rezultînd astfel problema la limită discretă

$$\begin{cases} Ax - qAf(P,x) = f(P,x) \\ x|_\delta = o \end{cases} \tag{2.21}$$

Cu ajutorul notaţiilor folosite mai sus rezultă sistemul de ecuaţii neliniare

$$Ax - \Phi(I - qA)\Phi x = o \tag{2.22}$$

cu A dat de (2.17), sistem care constituie ecuaţiile cu diferenţe finite cu corecţii de ordinul 1 cu un ordin de aproximare de $o(h^4)$, unde I este matricea unitate de dimensiune $m^2 x\, m^2$.

In continuare vom analiza existenţa şi unicitatea unei soluţii x^n a ecuaţiei (2.20) şi convergenţa şirului $\{x^n\}$ la soluţia problemei (2.13) în analogie cu teoria lui M.Lees [35(1)] pentru problema la limită bilocală slab neliniară. Ipoteza fundamentală pe care o vom face este că f satisface condiţia (2.15) considerată de J.Douglas [19(1)] pentru studiul metodei direcţiilor alternate, condiţie analoagă cu (2.3) pentru cazul unidimensional.

Vom da în prealabil două leme pregătitoare.

Lema 2.3. Presupunem că f satisface condiţia (2.15) şi fie h_o astfel încît

$$\eta < 2\pi^2(1 - h_o^2 \cdot \pi^2/12).$$

Dacă $h < h_o$ şi $x,y \in X_n$ verifică relaţiile

$$Ax = f(P,x) + q_1(P)$$
$$Ay = f(P,y) + q_2(P)$$

pentru orice P din d, atunci $\|x-y\|_d \leq k_o \|q_1-q_2\|_o$, unde

$$k_o = \frac{\sqrt{2}\,\pi}{2\pi^2(1-h_o^2\pi^2/12)-\eta}.$$

Demonstrația este întru totul analoagă cu demonstrația lemei c din [35(1)]. Singura deosebire care apare este aceea că cea mai mică valoare proprie a matricei A dată de (2.17) este $\hat{\lambda}_1 = 8h^{-2}\sin^2(\pi h/2)$ ([19(1)], [20(1)], [13(5)]) în loc de $\lambda_1 = 4h^{-2}\sin^2(\pi h/2)$ și prin urmare limitele lui λ_1 sînt

$$2\pi^2(1 - h^2\pi^2/12) \leq \lambda_1 \leq 2\pi^2$$

Din această lemă rezultă imediat următorul corolar :

Corolar. Dacă f satisface (2.15) problema discretă (2.19) are soluție unică.

Lema 2.4. Dacă f satisface condiția (2.15) și x este o soluție a ecuației (2.19) atunci

$$\|x\|_0 \leq (2\sqrt{2})^{-1}\|x\|_A \leq (2\sqrt{2})^{-1}k_0\|f_0\|_0 \quad (2.23)$$

unde $f_0 = f(P,0)$.

Demonstrație. Prima inegalitate rezultă direct din lema (2.2) pentru demonstrarea celei de a doua inegalități, aplicăm lema (2.3) în care $q_1(P) = 0$ și luăm $y = 0$. In acest caz $q_2(P) = -f(P,0) = -f_0$ astfel încît $\|x\|_A \leq k_0\|f_0\|_0$.

Teorema 2.5. Dacă f satisface condiția (2.15) atunci ecuațiile cu diferențe finite (2.19) sau, echivalent (2.20) au o soluție unică $x^n \in X_n$.

Demonstrație. Să notăm

$$\sigma(P,u) = \int_0^1 f_u(P, \overset{\circ}{\zeta}_u(P))d \quad .$$

Pentru fiecare $x \in X_0$ considerăm următoarea problemă la limită discretă

$$\begin{cases} \Delta y = \sigma(P,x)y + f_0(P) \quad , \quad P \in d \\ y|_\gamma = 0 \quad . \end{cases} \quad (2.24)$$

Vom arăta că (2.24) are o soluție unică. In acest scop considerăm următoarele probleme discrete cu valori inițiale .

$$Ay_1 = \sigma(P,x)y_1 + f_0(P)$$

$$y_1(t,o) = 0 \qquad y_1(o,s) = 0$$
$$y_1(t,h) = 0 \qquad y_1(l,s) = 0$$

$$Ay_{\ell k} = \sigma(P,x)y_{\ell k} \qquad k = \overline{1,m}$$

$$y_{\ell k}(t,o) = 0 \qquad y_{\ell k}(o,s) = 0$$
$$y_{\ell k}(t,h) = \alpha_{kj} \qquad y_{\ell k}(l,s) = 0$$

unde $\alpha_{kj} = 1$ pentru $k = j$ şi $\alpha_{kj} = 0$ pentru $k \neq j$. Este clar că aceste probleme au soluţii unice care se pot determina printr-un proces de calcul recursiv. Fie y_1 şi respectiv $y_{\ell k}$ soluţiile acestor probleme şi fie

$$z(t,s) = y_1(t,s) - \sum_{k=1}^{m} c_k y_{\ell k}(t,s) \qquad (t,s) \in \overline{d} \qquad (2.25)$$

Observăm că z satisface (2.24) (fără condiţia $z|_\delta = 0$) şi că $z(0,s) = 0$, $z(l,s) = 0$, $z(t,0) = 0$. Vom determina constantele c_k astfel încît $z(t,l) = 0$ şi atunci $z|_\delta = 0$, adică z va fi o soluţie a problemei (2.24). Din (2.25) şi din condiţia $z(t,l) = 0$, rezultă pentru c_k următorul sistem liniar

$$\begin{aligned} &c_1 y_{11}(t_1,l) + \ldots + c_m y_{1m}(t_1,l) = y_1(t_1,l) \\ &\vdots \\ &c_1 y_{11}(t_m,l) + \ldots + c_m y_{1m}(t_m,l) = y_1(t_m,l) \end{aligned} \qquad (2.26)$$

determinantul acestui sistem este diferit de zero. Într-adevăr dacă acest determinant ar fi zero, atunci ar exista constantele $c_1^o, \ldots, c_m^o$, nu toate nule, astfel încît

$$c_1^o y_{11}(t,l) + \ldots + c_m^o y_{1m}(t,l) = 0$$

Fie $z_1 = c_1^o y_{11} + \ldots + c_m^o y_{1m}$; este clar că z_1 verifică problema la limită

$$\begin{cases} Az_1 = \sigma(P,x)z_1 \\ z_1|_\delta = 0 \end{cases}$$

Pe de altă parte, dacă $c_k^o \neq 0$, atunci $z_1(t_k,h) = c_k^o \neq 0$ şi

această problemă nu mai admite soluţie unică, în contrazicere cu corolarul lemei 2.3.

Prin urmare, problema (2.24) admite soluţia (2.25) cu constantele c_k determinate de (2.26) şi conform corolarului amintit această soluţie este unică.

Fie acum $r = (2\sqrt{2})^{-1} k_0 \|f_0'\|_0$ şi $S = \{ x \mid \|x\|_d \leq r \} \subset X_n$. Definim operatorul $T : S \to X_n$ prin $Tx = y$ unde y este soluţia problemei (2.24). Conform lemei (2.4) $\|Ty\|_d = \|y\|_d \leq r$ şi deci T aplică sfera S în ea însăşi.

Să arătăm că T este un operator continuu. Fie $x_1, x_2 \in S$ şi y_1, y_2 valorile corespunzătoare ale lui T, deci

$$Ay_1 = \sigma(P, x_1) y_1 + f_0(P)$$

$$Ay_2 = \sigma(P, x_2) y_2 + f_0(P).$$

Dacă notăm $y = y_1 - y_2$ şi $H = \sigma(P, x_1) - \sigma(P, x_2)$ obţinem

$$Ay = \sigma(P, x_1) y_1 - \sigma(P, x_2) y_2 = \sigma(P, x_1) y + H y_2 .$$

De aici rezultă

$$\langle Ay, y \rangle = \langle \sigma(P, x_1) y, y \rangle + \langle H y_2, y \rangle$$

sau

$$\|y\|_A^2 = \langle \sigma(P, x_1) y, y \rangle + \langle H y_2, y \rangle$$

Cum $-\sigma(P, x_i) \leq \eta$ şi $\|y\|_0 \leq \lambda_1^{-1/2} \|y\|_A$ obţinem $\|y\|_A^2 \leq$

$$\leq \eta \|y\|_0^2 + \|H\|_d \|y_2\|_0 \|y\|_0 \leq \frac{\eta}{\sqrt{\lambda_1}} \|H\|_d \|y_2\|_0 \|y\|_A ,$$ aşadar

$$\|y\|_A \leq \frac{\|H\|_d \|y_2\|_0}{\sqrt{\lambda_1}(1 - \frac{\eta}{\lambda_1})} \leq k_0 \|H\|_d \|y_2\|$$

prin urmare, ţinînd cont de lema 2.2 şi de (2.16) rezultă

$$\|y\|_d \leq (2\sqrt{2})^{-1} \|y\|_A \leq (2\sqrt{2})^{-1} k_0 \|H\|_d \|y_2\|_0 \leq$$

$$(2\sqrt{2})^{-1} . h^{-1/2} k_0 r \|H\|_d$$

Fie acum $\varepsilon > 0$ arbitrar ; întrucît f_u este continuă,

există $\delta > 0$ astfel încît, dacă $\| x_1 - x_2 \| \leq \delta$ atunci

$$\| f_u(P, \xi x_1) - f(P, \xi x_2) \|_d \leq 2\sqrt{2}\, \varepsilon\, h^{1/2} (k_o r)^{-1}$$

pentru orice $\xi \in [0,1]$. Prin urmare $\|H\|_d \leq 2\sqrt{2}\, h^{1/2} (k_o r)^{-1}$ și $\|y\|_d \leq \varepsilon$, adică T este un operator continuu pe S cu valori în S. Conform teoremei lui Browder [60(1)] T are un punct fix $x^n \in S$.

Pe de altă parte x^n este o soluție a lui (2.19) întrucît

$$\sigma(P, x^n) x^n + f_o = f(P, x^n)$$

și ținînd cont de corolarul lemei (2.3) x^n este unică. Teorema este deci demonstrată.

Teorema 2.6. Dacă f satisface condiția (2.15), x^n este soluția ecuațiilor cu diferențe (2.20) iar u este o soluție a problemei (2.13), atunci

$$\| x^n - u \|_d \leq (2\sqrt{2})^{-1} \| x^n - u \|_A \leq \frac{h^2}{24\sqrt{2}}\, k_o \| M_4 \|_o$$

unde $M_4 = u_{t^4}(\xi_t, s) + u_{s^4}(t, \xi_s)$, $t-h < \xi_t < t+h$, $s-h < \xi_s < s+h$.

Demonstrație. Este suficient să demonstrăm inegalitatea a doua. Ținînd cont de (2.18), avem :

$$Au = \Delta u + \frac{h^2}{12} M_4 = f(P,u) + \frac{h^2}{12} M_4 .$$

Cum $Ax^n = f(P, x^n)$, inegalitatea cerută rezultă imediat din lema 2.3, în care $q_1 = h^2 M_4 / 12$ iar $q_2 = 0$.

Scanned with OKEN Scanner

CAPITOLUL 3

CONVERGENTA LOCALĂ A UNOR METODE ITERATIVE

In acest capitol se studiază convergenţa locală a unor metode iterative pentru ecuaţii neliniare în spaţii finit dimensionale sau în spaţii Hilbert reale şi determinarea iteraţiei de start. S-au considerat ecuaţii de forma (2.4),(2.5) şi (2.6) care provin din discretizarea unor ecuaţii diferenţiale neliniare (probleme cu condiţii la limită) sau de formă mai generală (2.12). Astfel, se generalizează [58(2)] o metodă iterativă directă clasică [5(1)] pentru ecuaţii de forma (2.12), se studiază [58(7)] o variantă a metodei direcţiilor alternate pentru cazul neliniar considerată de Kellogg [30(1)] şi se generalizează [59(2)] o metodă de tip gradient considerată de Altman [1(1),(2)]. In cazuri particulare se obţin unele metode iterative cunoscute, în general cu condiţii de convergenţă mai slabe, ceea ce corespunde cu faptul că condiţiile considerate asigură numai o convergenţă locală.

Menţionăm că probleme rezolvării numerice a ecuaţiilor care provin din discretizarea unor ecuaţii diferenţiele neliniare, în mod special a ecuaţiilor cu diferenţe corespunzătoare problemelor la limită de tip eliptic slab neliniare au fost considerate de numeroşi autori. Astfel Bers [7(1)], Ortega şi Rockoff [48(1)] iar mai recent Gergei [22(1)] au considerat metoda Gauss-Seidel neliniară (metoda SOR - Successive Over Relaxation - cu parametru de relaxare 1) Douglas [19(1)] şi Gunn [24 (1)] au studiat metoda direcţiilor alternante a lui Peaceman-

Rachford [54(1)] pentru ecuaţii eliptice neliniare iar Scorobogatko şi Mamunya [66(1)] au considerat o metodă specială de aproximaţii succesive.

3.1. Introducere

In acest paragraf introductiv vom prezenta unele leme simple de analiză pe spaţii Banach şi două teoreme de convergenţă locală. De asemenea vom preciza noţiunea de operator monoton şi vom prezenta teorema de surjectivitate a lui Minty.

3.1.1. Leme preliminare.

Fie B un spaţiu Banach real, $L(B)$ spaţiul Banach al operatorilor liniari şi mărginiţi de la B în B şi A un operator (neliniar) definit pe B cu valori în $L(B)$.

Lema 3.1. Presupunem că A este diferenţiabil Fréchet în $x^0 \in B$ şi fie y un element arbitrar din B. Atunci operatorul $T:B \to B$ definit de $Tx = A(x)y$ este diferenţiabil Fréchet în x^0.

Demonstraţie. Rezultă imediat din diferenţiabilitatea Fréchet a operatorului A în x^0 şi din faptul că operatorul $A_1: B \to B$ definit de $A_1 h = A'(x^0)hy$ este liniar. In continuare vom nota derivata Fréchet a operatorului T în punctul x^0 cu $[A(x)y]'_{x^0}$ deci

$$T'(x^0)h = [A(x)y]'_{x^0} h = A'(x^0)\, hy$$

Lema 3.2. Dacă operatorii $A:B \to L(B)$ şi $F:B \to B$ sînt diferenţiabili Fréchet în x^0, atunci operatorul $T:B \to B$ definit de $Tx = A(x)Fx$ este de asemenea diferenţiabil Fréchet în x^0 şi

$$T'(x^0) = [A(x)Fx^0]'_{x^0} + A(x^0)F'(x^0) \qquad (3.1)$$

Demonstraţie. Deoarece operatorii A, F şi $A(\cdot)Fx^0$ sînt diferenţiabili Fréchet în x^0, avem :

$$\|Fx - Fx^0 - F'(x^0)(x-x^0)\| \leq \varepsilon \|x-x^0\|$$

$$\|A(x)-A(x^0)-A'(x^0)(x-x^0)\| \leq \varepsilon \|x-x^0\|$$

$$\|A(x)Fx^0-A(x^0)Fx^0-[A(x)Fx^0]'_{x^0}(x-x^0)\| \leq \varepsilon \|x-x^0\| .$$

unde ε este un număr pozitiv dat iar x aparţine sferei centrate în x^0. Din primele două relaţii rezultă

$$\|Fx-Fx^0\| \leq (\varepsilon + \|F'(x^0)\|)\|x-x^0\|$$

$$\|A(x)-A(x^0)\| \leq (\varepsilon + \|A'(x^0)\|)\|x-x^0\|$$

Printr-un calcul simplu se obţine :

$$\|Tx-Tx^0-T'(x^0)(x-x^0)\| \leq [\varepsilon + \varepsilon\|Ax^0\| + (\varepsilon + \|F'(x^0)\|)(\varepsilon + \|A'(x^0)\|)\|x-x^0\|]\|x-x^0\|$$

unde $T(x^0)$ este dat de (3.1).

Observaţia 3.1. Formula (3.1) generalizează formula binecunoscută de derivare a unui produs.

3.1.2. Convergenţa locală. Dacă x^0 aparţine lui B şi r este un număr real pozitiv, vom nota cu $S(x^0,r)$ sfera deschisă centrată în x^0 şi de rază r, $S(x^0,r)=\{x \in B, \|x-x^0\| < r\}$; prin $\bar{S}(x^0,r)=\{x \in B, \|x-x^0\| \leq r\}$ notăm sfera închisă centrată în x^0 şi de rază r. Fie D o mulţime deschisă $D \subset B$ şi $T:D \to B$ un operator neliniar. Vom considera în continuare şirul $\{x^k\} \subset B$ definit de procesul iterativ,

$$x^{k+1} = Tx^k \qquad (3.2)$$

Definiţia 3.1.

Vom spune că $x^* \in D$ este un punct de atracţie pentru iteraţia (3.2) dacă există o sferă $S(x^*,r) \subset D$ astfel încît pentru orice $x^0 \in S(x^*,r)$ şirul $\{x^k\}$ generat de (3.2) să aparţină lui D şi $x^k \to x^*$ cînd $k \to \infty$.

Dacă x^* este un punct de atracţie pentru iteraţia (3.2) vom spune că şirul $\{x^k\}$ converge local la x^* sau că metoda asigură o convergenţă locală. In aplicaţii concrete, convergenţa

unui şir generat de o metodă local convergentă se asigură prin alegerea convenabilă a iteraţiei de start.

Vom nota cu $Q_t(x^k)$, $R_t(x^k)$, Q şi respectiv R - factorii de convergenţă corespunzători şirului $\{x^k\}$ [49(1)]. Dacă $A \in L(B)$ vom nota cu $\rho(A)$ raza spectrală a operatorului A. Următoarea teoremă dată de Ostrowski [50(1)] constituie o condiţie suficientă de convergenţă locală :

Teorema 3.1. Fie $T:D \subset B \to B$ un operator neliniar şi $x^* \in D$ un punct fix al lui T. Presupunem că T este diferenţiabil Fréchet în x^* şi că $\rho(T'(x^*)) = \sigma < 1$. Atunci x^* este punct de atracţie pentru iteraţia (3.2) şi $R_t\{x^k\} \leq Q_t\{x^k\} \leq \sigma$.

Condiţia $\sigma < 1$, greu de verificat în cazuri concrete, se poate înlocui cu condiţia $\|T'(x^*)\| < 1$, ţinînd cont că pentru orice normă $\rho(T'(x^*)) \leq \|T'(x^*)\|$. In cazul unui spaţiu Banach finit dimensional, dacă $\sigma > 0$ atunci convergenţa este liniară (mai exact, q şi r - ordinele de convergenţă sînt 1).

3.1.3. Operatori cvasi-neexpansivi

Definiţia 3.2. Vom spune că operatorul $T:D \subset B \to B$ este cvasi-neexpansiv pe D în raport cu $x^* \in D$ dacă

$$\|Tx - x^*\| < \|x - x^*\| \qquad \forall\, x \in D \qquad (3.3)$$

De obicei T se consideră cvasi-neexpansiv în raport cu mulţimea punctelor fixe ale lui T în D ; de asemenea în (3.3) inegalitatea strictă este uneori înlocuită cu semnul $\leq$. In continuare vom considera proprietatea de cvasi-neexpansivitate în raport cu un element $x^* \in D$ (în general punct fix al lui T) şi vom menţiona punctul relativ la care T este cvasi-neexpansiv.

Să observăm că dacă x^* este punctul fix al lui T şi (3.3) are loc, atunci x^* este punctul fix unic pentru T în D.

In cazul unui spaţiu de dimensiune finită pe $\mathbb{R}^n$ are loc

următoarea proprietate de convergenţă locală pentru operatori cvasi-neexpansivi :

Lema 3.3. Fie D o mulţime deschisă din $\mathbb{R}^n$, $x^* \in D$ şi fie $T: D \setminus \{x^*\} \longrightarrow \mathbb{R}^n$ un operator continuu pe $D \setminus \{x^*\}$. Presupunem că T este cvasi-neexpansiv pe $S = \bar{S}(x^*, r) \setminus \{x^*\}$, unde $\bar{S}(x^*, r) \subset D$. Atunci $x^k = T^k x^0 \longrightarrow x^*$ cînd $k \longrightarrow \infty$ pentru orice $x^0 \in S$.

Demonstraţie.

Ţinînd cont de (3.3) rezultă $\|x^{k+1} - x^*\| < \|x^k - x^*\|$ pentru orice k natural, astfel încît şirul $\{x^k\}$ rămîne în S şi $\|x^k - x^*\| \longrightarrow a$ cînd $k \longrightarrow \infty$. Să presupunem că $a > 0$. Fie atunci $\{x^{k_j}\}$ un subşir convergent al şirului $\{x^k\}$, $x^{k_j} \longrightarrow y^*$ cînd $k_j \longrightarrow \infty$. Presupunem că $y^* \neq x^*$. Evident $\|y^* - x^*\| = a$ şi din continuitatea lui T rezultă :

$$\lim_{k_j \to \infty} x^{k_j+1} = \lim_{k_j \to \infty} T x^{k_j} = T y^* ,$$

deci

$$a = \lim_{k_j \to \infty} \|x^{k_j+1} - x^*\| = \|Ty^* - x^*\| < \|y^* - x^*\| = a$$

ceea ce este o contradicţie, deci $y^* = x^*$. Pentru $y^* = x^*$ avem

$$\lim_{k_j \to \infty} \|x^{k_j} - x^*\| = 0$$

şi prin urmare

$$\lim_{k \to \infty} \|x^k - x^*\| = 0$$

Lema este demonstrată.

Această lemă constituie o generalizare a unui rezultat al lui Tricomi [71(1)] relativ la convergenţa locală a iteraţiei $x^{k+1} = f(x^k)$, f fiind o funcţie reală definită pe un interval (a,b). Sub o formă puţin modificată lema 3.3 apare în [56(1)] şi este utilizată pentru demonstrarea convergenţei unor şiruri de aproximaţii succesive în spaţii metrice. In afara condiţiei de cvasi-neexpansivitate (considerată în raport cu mulţimea

punctelor fixe $F(T)$) se mai impune şi condiţia ca să existe un subşir $\{x^{k_j}\}$ a lui $\{x^k\}$ astfel încît $d(x^{k_j}, F(T)) \longrightarrow 0$ cînd $k_j \longrightarrow \infty$ unde prin $d(x^{k_j}, F(T))$ se înţelege distanţa de la punctul x^{k_j} la mulţimea $F(T)$. Menţionăm că condiţia ca T să fie continuu pe D se poate înlocui cu condiţia mai slabă ca $F(T)$ să fie o mulţime închisă [41(1),(2)].

Proprietatea de cvasi-neexpansivitate a fost utilizată în mai multe lucrări [56(1)], [18(1)], [41(4)], pentru studiul convergenţei unor metode iterative. De exemplu în [41(4)] se arată că funcţiile de iteraţie din metoda Newton şi din metoda gradientului, sînt cvasi-neexpansive, convergenţa acestor metode rezultînd pe această bază. In cazul funcţiilor reale, Stepleman [68(1)] a arătat că condiţia necesară şi suficientă ca şirul $\{x^k\}$ generat de $x^{k+1} = f(x^k)$ să fie convergent este ca f^2 să fie cvasi-neexpansivă pe un anumit interval $(x^* - \varepsilon, x^* + \varepsilon)$.

3.1.4. Operatori monotoni. Fie H un spaţiu Hilbert real (produs scalar $\langle \cdot, \cdot \rangle$ şi normă $\| \cdot \|$) şi fie $T: H \longrightarrow H$ un operator neliniar.

Definiţia 3.3. Vom spune că operatorul T este strict monoton pe H dacă există o constantă $c > 0$ astfel încît :

$$\langle Ty - Tx, y - x \rangle \geq c \| y - x \|^2 \quad \forall x, y \in H \qquad (3.4)$$

Dacă $c = 0$, T se numeşte monoton.

Dacă T este diferenţiabil Fréchet în $x \in H$ şi $T'(x)$ este pozitiv definit atunci T este strict monoton pe o anumită sferă centrată în x.

Relativ la operatorii monotoni, menţionăm următoarea teoremă de surjectivitate a lui Minty [42(1)] pe care o vom utiliza în continuare la analiza convergenţei unei variante a metodei ADI pentru cazul neliniar :

Teorema 3.2. Dacă T este continuu şi strict monoton pe H

atunci T este surjectiv.

Noţiunea de monotonie în sensul definiţiei 3.3 a fost introdusă de Minty [42(1)] în cadrul mai general al spaţiilor Hilbert complexe (produsul scalar din stînga relaţiei (3.4) se înlocuieşte în cazul spaţiilor complexe, cu partea reală). Teorema 3.2 reprezintă unul din rezultatele de bază din teoria operatorilor monotoni. Menţionăm că teoria operatorilor monotoni joacă un rol deosebit în studiul ecuaţiilor neliniare, atît în ce priveşte existenţa soluţiilor ecuaţiilor cît şi convergenţa unor metode iterative [8(1)], [72(1)], [65(1)].

3.2. Determinarea iteraţiei de start pentru metode iterative.

Fie operatorul neliniar $F \in C(D, \mathbb{R}^n)$, unde D este o mulţime deschisă şi mărginită din $\mathbb{R}^n$ şi ecuaţia neliniară

$$F(x) = \theta \qquad (3.5)$$

unde θ este zeroul din $\mathbb{R}^n$. Pentru convergenţa locală a metodelor iterative este în general suficient ca iteraţia de start $x^0 \in S(x^*, r)$, unde prin S s-a notat sfera de convergenţă a metodei, r este raza de convergenţă şi x^* soluţia ecuaţiei (3.5).

3.2.1. Căutarea iteraţiei de start în nodurile unei reţele de puncte. Atunci cînd în practică nu se poate da o evaluare suficient de bună lui x^* pentru a folosi această valoare ca iteraţie de start putem folosi un subprogram care inhibă mesajul de eroare "depăşire flotantă" (în acest caz procesul iterativ ales este divergent) şi relansează procesul cu o nouă iteraţie de start. Alegerea noii valori se poate face printr-un algoritm de parcurgere a vîrfurilor unui cub n-dimensional ale cărui laturi cresc pe rînd pe cîte o direcţie. In exemplul prezentat se parcurge reţeaua de puncte din spaţiul 3-dimensional pentru a găsi iteraţia de start x^0 care asigură convergenţa metodei Newton.

Scanned with OKEN Scanner

Fie sistemul

$$\begin{cases} 4\sqrt{x^2+1}/x = 5\sqrt{y^2+1}/y = 6\sqrt{z^2+1}/z \\ x + y + z = xyz \end{cases}$$

STARTED

Ø - . 7ØØØØØ .ØØØØØØ 6.5ØØØØ

SIRUL DIVERGE, EROAREA 1Ø8 A FOST INHIBATA LA PASUL Ø

Ø -.7ØØØØØ .ØØØØØØ 8.ØØØØØ

SIRUL DIVERGE, EROAREA 1Ø8 A FOST INHIBATA LA PASUL Ø

.

.

.

Ø .8ØØØØØ .ØØØØØØ 3.5ØØØØ

SIRUL DIVERGE, EROAREA 108 A FOST INHIBATA LA PASUL 0

Ø .8ØØØØØ 1.5ØØØØ 6.5ØØØØ

SOLUTIA SISTEMULUI

19 .881917 1.46986 7.93765

Pentru ca metoda să fie eficientă rețeaua de puncte ar trebui să fie aleasă în așa fel ca distanța dintre două puncte alăturate să nu fie mai mare ca $2r$, unde r este raza de convergență a metodei, dar determinarea lui r este o intreprindere dificilă.

3.2.2. Căutarea aleatoare a iterației de start. Parcurgerea sistematică a nodurilor unei rețele poate fi înlocuită cu generarea aleatoare a unor puncte $x^0 \in D$ pentru care se verifică dacă metoda iterativă aleasă este convergentă sau nu. Această metodă elimină neajunsul de la 3.2.1, dar s-ar putea să fie laborioasă.

3.2.3. Căutarea iterației de start cu ajutorul gradului topologic. Fie operatorul neliniar $F \in C^1(D,\mathbb{R}^n) \cap C(D,\mathbb{R}^n)$ și notăm cu F' derivata Gateaux a lui F pentru care considerăm reprezentarea sub forma :

$$F'(X) = (\partial f_i(x)/\partial x_j) \qquad i = \overline{1,n} \ , \ j = \overline{1,n}$$

Scanned with OKEN Scanner

unde $F = (f_1, f_2, \ldots, f_n)$ cu $f_i : D \longrightarrow \mathbb{R}$ pentru $\forall\, i = \overline{1,n}$.

Definiţia 3.4. Un element $x \in D$ se numeşte regulär dacă şi numai dacă $\det F'(x) \neq 0$ (unde s-a notat cu "det" determinantul matricii).

Pentru ecuaţia

$$F(x) = y \qquad (3.6)$$

unde $y \notin F(\partial D)$ conform teoremei funcţiei inverse rezultă că $F^{-1}(y)$ este o mulţime finită, dacă toate elementele $x \in F^{-1}(y)$ sînt regulare [51(1)].

Definiţia 3.5. (Gradul topologic al lui Brouwer)

i) Dacă $F \in C^1(D,\mathbb{R}^n) \cap C(\bar{D},\mathbb{R}^n)$, $y \notin F(\partial D)$ şi $\det F'(x) \neq 0$ pentru orice $x \in F^{-1}(y)$, atunci prin definiţie gradul topologic al lui F relativ la domeniul D şi punctul y este

$$d(F,D,y) = \sum_{x \in F^{-1}(y)} \operatorname{sgn}(\det F'(x))$$

ii) Dacă $F \in C^1(D,\mathbb{R}^n) \cap C(\bar{D},\mathbb{R}^n)$, $y \notin F(\partial D)$ şi dacă $F_k \in C^1(D,\mathbb{R}^n) \cap C(\bar{D},\mathbb{R}^n)$ pentru $k \in \mathbb{N}$ este un şir de funcţii ce converge uniform către F şi $y \notin F_k(\partial D)$ pentru orice $k \in \mathbb{N}$ atunci

$$d(F,D,y) = \lim_{k \to \infty} d(F_k,D,y).$$

iii) Dacă $F \in C^1(D,\mathbb{R}^n) \cap C(\bar{D},\mathbb{R}^n)$, $y \notin F(\partial D)$ şi există cel puţin un $x \in F^{-1}(y)$ astfel încît $\det F'(x) = 0$, (adică y este o valoare singulară a lui F) şi $y_k \notin F(\partial D)$, $k \in \mathbb{N}$ un şir de valori regulare a lui F, cu $y_k \longrightarrow y$ cînd $k \longrightarrow \infty$, atunci

$$d(F,D,y) = \lim_{k \to \infty} d(F,D,y_k),$$

In continuare prezentăm o serie de teoreme fără demonstraţie, indicînd sursa bibliografică, spre a justifica cele afirmate. Următoarele 3 leme justifică definiţia gradului topologic al lui Brouwer.

Scanned with OKEN Scanner

Lema 3.4. Fie $F \in C^1(D,\mathbb{R}^n)$ şi G o mulţime deschisă astfel încît $\bar{G} \subset D$ şi notăm cu

$$B = \{ x \in G \mid \det F'(x) \neq 0 \} .$$

Atunci măsura Lebesque n-dimensională a lui $F(B)$ este zero [51(1)] .

Lema 3.5. Fie $F \in C^1(D,\mathbb{R}^n) \cap C(\bar{D},\mathbb{R}^n)$, $y,z \notin F(\partial D)$ valori regulare pentru F. Atunci dacă y şi z sînt suficient de apropiate, avem :

$$d(F,D,y) = d(F,D,z) \quad [61(1)] .$$

Lema 3.6. Fie $F,H \in C^1(D,\mathbb{R}^n) \cap C(\bar{D},\mathbb{R}^n)$, $y \notin F(\partial D)$ şi $y \notin H(\partial D)$, y valoare regulară pentru F şi H. Dacă F şi H sînt suficient de apropiate în $C^1(D,\mathbb{R}^n)$, atunci :

$$d(F,D,y) = d(H,D,y) \quad [61(1)]$$

In cele ce urmează precizăm cîteva proprietăţi ale gradului topologic al lui Brouwer, care justifică cele afirmate în algoritmii de mai jos.

Teorema 3.3. (Dependenţa de valorile de pe frontieră). Dacă $F,H \in C(\bar{D},\mathbb{R}^n)$ $F\big|_{\partial\bar{D}} = H\big|_{\partial\bar{D}}$, $d(F,D,y)$ şi $d(H,D,y)$ sînt definite, atunci

$$d(F,D,y) = d(H,D,y) \quad [61(1)] .$$

Notăm norma euclidiană din $\mathbb{R}^n$ cu $\|X\| = (\sum_{i=1}^{n} x_i^2)^{1/2}$.

Teorema 3.4.(Formula integrală a gradului topologic)(Kroneker). Fie $F \in C(D,\mathbb{R}^n)$ şi ecuaţia (3.5) iar $\theta \in F(\partial D)$, atunci pentru $n > 1$ avem

$$d(F,D,\theta) = Q \int_{\partial D} w . \det(F, \frac{\partial F}{\partial u^1}, \frac{\partial F}{\partial u^2}, \ldots, \frac{\partial F}{\partial u^n}) du^1 du^2 \ldots du^n$$

iar pentru $n = 1$

$$d(F,D,\theta) = (F(b)/|F(b)| - F(a)/|F(a)|)/2$$

unde $Q = 1/S_{n-1}$ şi S_{n-1} este volumul n-1 dimensional al suprafeţei sferei de dimensiune n-1 şi de rază 1, iar $w = 1/\|F\|^n$, $a = \inf D$, $b = \sup D$ [66(1)] .

Teorema 3.5. (Proprietatea relativă la descompunerea domeniului). Fie $D_i \subset D$, $i = \overline{1,k}$, submulţimi deschise şi disjuncte ale lui D, iar $F \in C(D,R^n)$. Dacă avem ecuaţia (3.6) şi $F(x) \neq y$ pentru orice $x \in \bar{D} \setminus \bigcup_{i=1}^{k} D_i$, atunci

$$d(F,D,y) = \sum_{i=1}^{k} d(F,D_i,y).$$

Demonstraţia este imediată din definiţia gradului topologic al lui Brouwer.

Teorema 3.6. (Teorema de existenţă). Dacă $d(F,D,y)$ este definit şi diferit de zero rezultă că ecuaţia (3.6) are cel puţin o soluţie $x^* \in D$.

Demonstraţia este imediată din definiţia gradului topologic al lui Brouwer.

Corolarul 1. Dacă ecuaţia (3.6) are într-o mulţime $G \subset D$ o unică soluţie x^*, iar $G = G1 \cup G2$ şi $\mathring{G}1 \cap \mathring{G}2 = \emptyset$ atunci :

- dacă $x^* \in G1$ rezultă $d(F,G1,y) = \pm 1$ şi $d(F,G2,y) = 0$
- dacă $x^* \in G2$ rezultă $d(F,GL,y) = 0$ şi $d(F,G2,y) = \pm 1$.

Fie $X_1, X_2, \ldots, X_q$, $q+1$ puncte din R^n, unde $q \leq n$ astfel încît punctele $X_1 - X_o$, $X_2 - X_o, \ldots, X_q - X_o$ sînt liniar independente. Spunem că q - simplexul format din aceste puncte definit de mulţimea :

$$[X_o, X_1, \ldots, X_q] = \{X \in R^n,\ X = p_o X_o + p_1 X_1 + \ldots + p_q X_q\} \text{ unde} \tag{3.7}$$

$$p_o + p_1 + \ldots + p_q = 1 \text{ şi } p_i \geq 0 \text{ pentru orice } i = \overline{0,q}$$

are orientare pozitivă (sau negativă) în R^n dacă :

$$\det(Z_o, Z_1, \ldots, Z_q) > 0 \quad (\text{sau } < 0)$$

unde $Z_i = (1, X_i)$.

Poliedrul convex $P \in R^n$, definit prin cele $m+1$ vîrfuri ale sale ($m > n$) $X_o, X_1, \ldots, X_m$, poate fi descompus într-o sumă de p n-simplexe

$$P = \sum_{k=1}^{p} S_k = \sum_{k=1}^{p} [X_0^{(k)}, X_1^{(k)}, \ldots, X_n^{(k)}] \qquad (3.8)$$

astfel încît n-simplexele să fie disjuncte două cîte două. Această descompunere se realizează cu algoritmul 3.1. Frontiera n-simplexului S_k este reprezentată de suma a (n+1) simplexe orientate de dimensiune (n-1) dat de

$$f(S_k)=\sum_{i=0}^{n} (-1)^i [X_0^{(k)}, X_1^{(k)}, \ldots, X_{i-1}^{(k)}, X_{i+1}^{(k)}, \ldots, X_n^{(k)}].$$

Atunci evident că

$$f(P)=\sum_{k=1}^{p} t_k \sum_{i=0}^{n} (-1)^i [X_0^{(k)}, X_1^{(k)}, \ldots, X_{i-1}^{(k)}, X_{i+1}^{(k)}, \ldots, X_n^{(k)}] \qquad (3.9)$$

cu $t_k \in \{1,-1\}$ orientarea simplexului S_k și frontiera unei mulțimi mărginite.

Considerăm următoarea generalizare a funcției sgn: $\mathbb{R} \longrightarrow \{-1,0,1\}$

$$Sgn: \mathbb{R}^n \longrightarrow \mathbb{R}^n,\ Sgn(X)=(sgn(X_1), sgn(X_2), \ldots, sgn(X_n)).$$

Fie formula următoare :

$$g_p(F,P,\theta)=\Big(\sum_{k=1}^{p} t_k \sum_{i=0}^{k} (-1)^i q_i\Big)/2^n.n! \quad , \text{ unde} \qquad (3.10)$$

$$q_i = \det(Sgn\ F(X_0^k), \ldots, Sgn\ F(X_{i-1}^k), Sgn\ F(X_{i+1}^k), \ldots, Sgn\ F(X_n^k))$$

Teorema 3.7. Fie $F \in C(D,\mathbb{R}^n)$, $D \in \mathbb{R}^n$ și ecuația (3.5) astfel încît $F(x) \neq \theta$ pentru orice $x \in f(P)$. Pentru un $p \in \mathbb{N}$ suficient de mare dar finit, avem că :

$$g_p(F,P,\theta) = d(F,P,\theta).$$

Teorema este o consecință a teoremelor 1, 2 și a faptului că $d(F,P,\theta) \in \mathbb{Z}$ [67(1)].

Algoritmul 3.1. (de descompunere a poliedrului P)

Notăm cu M mulțimea $\{X_0, X_1, \ldots, X_m\}$ a vîrfurilor poliedrului convex P. Fie $X_{i_1}, X_{i_2}, \ldots, X_{i_n}$ primele n vîrfuri din M și funcția $H: \mathbb{R}^{n+1} \longrightarrow \mathbb{R}$,

$$H(X,X_{i1},X_{i2},\dots,X_{in}) = \det(Z,Z_{i1},Z_{i2},\dots,Z_{in})$$

şi $Z = (X,1)$, $Z_{ik} = (X_{ik},1)$.

Pasul 1 - Dacă există $Y_o \in M \setminus \{X_{i1},X_{i2},\dots,X_{in}\}$ astfel încît $H(Y_o) < 0$ sau (> 0) şi dacă pentru orice Y_i , $Y_i \neq Y_o$, $Y_i \in M \setminus \{X_{i1},X_{i2},\dots,X_{in}\}$, $H(Y_i) < 0$ (sau > 0), atunci $[Y_o,X_{i1},X_{i2},\dots,X_{in}]$ este un n-simplex din descompunerea poliedrului P. (Urmează pasul 2).

- In caz contrar se consideră alte n vîrfuri din M din cele C_{m+1}^{n} posibilităţi şi se repetă pasul 1.

Pasul 2. Se elimină Y_o din mulţimea M şi se face $m = m-1$.

- Dacă $m \geq n$ se continuă cu pasul 1

- Dacă $m < n$ procesul de calcul se opreşte.

Observaţia 3.2. Convexitatea poliedrului P, păstrarea sensului şi a ordinii de parcurgere a vîrfurilor la introducerea lor, asigură că algoritmul 3.1 determină întotdeauna în cadrul pasului 1 un n-simplex de descompunere. Astfel avem descompunerea poliedrului P în sumă de p-simplexe ca în (3.8).

Dacă $\theta \notin F(f(P))$ atunci putem calcula gradul topologic $d(F,P,\theta)$ cu ajutorul următorului algoritm :

Algoritmul 3.2. (Stenger)

Pasul 1. Calculăm orientarea fiecărui n-simplex $S_k = [X_o^{(k)},X_1^{(k)},\dots,X_n^{(k)}]$, $k = \overline{1,p}$ cu formula :

$$t_k = \operatorname{sgn} H(X_o^{(k)},X_1^{(k)},\dots,X_n^{(k)}).$$

Pasul 2. Calculăm cu formula (3.10) pe $\varepsilon_p(F,P,\theta)$.

- Dacă $\varepsilon_p(F,P,\theta) \in Z$ şi este egal cu cel calculat în ciclul anterior al algoritmului (în primul ciclu al algoritmului această condiţie evident nu este îndeplinită), adică :

$$\varepsilon_p(F,P,\theta) = \varepsilon_{p/2}(F,P,\theta) \quad , \quad \text{atunci}$$

$$d(F,P,\theta) = \varepsilon_p(F,P,\theta)$$

şi procesul de calcul se opreşte.

- Dacă $g_p(F,P,\theta) \notin \mathbb{Z}$ sau nu este egal cu cel calculat în ciclul anterior, se continuă cu pasul 3

Pasul 3. Pentru fiecare n-simplex S_k $k = \overline{1,p}$ determinăm vectorul de lungime maximă. Fie el $\overline{X_i^{(k)} X_j^{(k)}}$ $0 \leq i < j \leq n$. Considerăm punctul :

$$Y = (X_i^{(k)} + X_j^{(k)})/2.$$

Simplexul de dimensiune n S_k se va putea scrie ca sumă de două simplexe de dimensiune n după cum urmează : $S_k = S'_k + S_{k+p}$, unde

$$S'_k = [X_0^{(k)}, X_1^{(k)}, \ldots, X_{i-1}^{(k)}, Y, X_{i+1}^{(k)}, \ldots, X_n^{(k)}]$$

$$S_{k+p} = [X_0^{(k)}, X_1^{(k)}, \ldots, X_{j-1}^{(k)}, Y, X_{j+1}^{(k)}, \ldots, X_n^{(k)}].$$

După ce s-a terminat descompunerea tuturor n-simplexelor S_k facem $p = 2p$ şi reluăm algoritmul cu pasul 1.

Algoritmul 3.3. (de determinare a iteraţiei de start).

Fie, P un poliedru convex din $\mathbb{R}^n$ în care ecuaţia (3.5) are o unică soluţie care aparţine lui P. De asemenea fie r precizia cu care determinăm iteraţia de start x^0 (r poate fi raza de convergenţă a metodei iterative cu care dorim să determinăm o soluţie de precizie a ecuaţiei (3.5)). Fie descompunerea poliedrului P în sumă de n-simplexe ca în (3.8).

Pasul 1. Calculăm $d(F,S_k,\theta)$ pentru fiecare k , $k = \overline{1,p}$.

- Dacă există $j \in \{1,2,\ldots,p\}$ astfel încît $d(F,S_j,\theta) = \pm 1$ iar $d(F,S_k,\theta) = 0$ pentru $k \in \{1,2,\ldots,p\}$, $k \neq j$ atunci urmează pasul 2.

- In caz contrar P nu conţine o soluţie unică a ecuaţiei (2.1), algoritmul se opreşte. Trebuie reconsiderat poliedrul P.

Pasul 2. Determinăm vectorul de lungime maximă în simplexul S_j. Fie el $\overline{X_i^{(j)} X_k^{(j)}}$ unde $0 \leq i < k \leq n$ şi punctul Y dat de formula

$$Y = (X_i^{(j)} + X_k^{(j)})/2$$

Simplexul de dimensiune n S_j se va putea scrie ca sumă de două simplexe de dimensiune n după cum urmează

$$S_j = S_{j1} + S_{j2} \text{ , unde}$$

$$S_{j1} = [X_o^{(j)}, X_1^{(j)}, \ldots, X_{i-1}^{(j)}, Y, X_{i+1}^{(j)}, \ldots, X_n^{(j)}]$$

$$S_{j2} = [X_o^{(j)}, X_1^{(j)}, \ldots, X_{k-1}^{(j)}, Y, X_{k+1}^{(j)}, \ldots, X_n^{(j)}]$$

Pasul 3. Calculăm $d(F, S_{j1}, \Theta)$ și $d(F, S_{j2}, \Theta)$.

- Dacă $d(F, S_{j1}, \Theta) = \pm 1$ și $d(F, S_{j2}, \Theta) = 0$ atunci $S_j = S_{j1}$. Se continuă cu pasul 4.
- Dacă $d(F, S_{j1}, \Theta) = 0$ și $d(F, S_{j2}, \Theta) = \pm 1$ atunci $S_j = S_{j2}$. Se trece la pasul 4.
- Dacă $d(F, S_{j1}, \Theta) = d(F, S_{j2}, \Theta)$ atunci $Y = 51 \, m(X_i^{(j)} + X_k^{(j)})/100$ și se reia pasul 3.

Pasul 4. Se calculează centrul de greutate al n-simplexului S_j cu formula :

$$G = (X_o^{(j)} + X_1^{(j)} + \ldots + X_n^{(j)})/(n+1).$$

Fie $e = \max \{ |GX_i^{(j)}|, \; i = \overline{0,n} \}$.

- Dacă $e < r$ atunci $x^0 = G$ este iterația de start căutată și algoritmul se oprește.
- Dacă $e \geq r$ reluăm algoritmul cu pasul 2.

Exemple.

1. Fie sistemul neliniar

$$\begin{cases} -\sin(e^x) + y - 1 = 0 \\ \operatorname{ch}(\sqrt{x}) - \ln(y^3) = 0. \end{cases}$$

Ecuația are o unică soluție în poliedrul de dimensiune 2 (patrulater) cu vîrfurile (0.1,0.1),(0.1, 3.),(4.,2.),(2.,0.1). Cu ajutorul a 3 programe ~~și a unei proceduri~~ prezentate mai jos s-au determinat cele două 2-simplexe (triunghiuri) de descom-

punere ale patrulaterului şi apoi şirul de 2-simplexe (triunghiuri) şi centrele lor de greutate care conţin unica soluţie în acest poliedru de dimensiune 2. Căutarea se opreşte dacă maximul distanţelor de la centrul de greutate la vîrfurile 2-simplexului este mai mic decît precizia impusă r (r = 0,1).

Simplexul 1 (Ø.1ØØ,Ø.1ØØ) (Ø.1ØØ,3.ØØØ) (4.ØØØ,2.ØØØ)

Simplexul 2 (Ø.1ØØ,Ø.1ØØ) (Ø.1ØØ,3.ØØØ) (2.Ø5Ø,1.Ø5Ø)

.
.
.

Simplexul 11 (Ø.832,1.725) (Ø.954,1.6Ø4) (Ø.832,1.655)

Centrul de greutate	Precizia
G1=(1.4ØØ,1.7ØØ)	2.617
G2=(Ø.75Ø,1.383)	1.742
.	
.	
.	
G11=(Ø.873,1.655)	Ø.Ø95

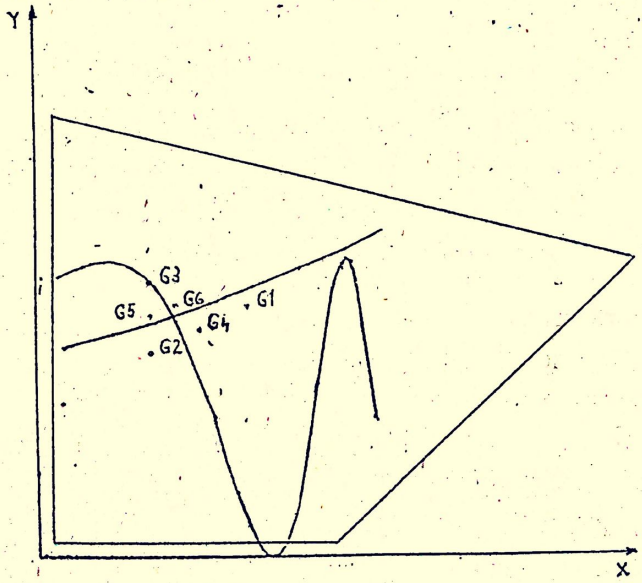


Fig.1.

2. Fie sistemul neliniar

$$\begin{cases} \sin x + x/6.3219 + y^3 - 2 = 0 \\ \sqrt{x} - \ln y + z^2 - 2 = 0 \\ y^2 - 2e^{z-1} + 1 = 0 \end{cases}$$

S-a pornit determinarea iteraţiei de start cu poliedrul de dimensiune 3 (paralelipiped dreptunghic) $[0,2]\times[0.1,2]\times[0,2]$. Precizia r de determinare a iteraţiei de start s-a considerat 0.1.

Centrul de greutate	Precizia
G1=(Ø.8ØØ,Ø.85Ø,1.25Ø)	1.662
.	
.	
.	
G18=(1.ØØ2,Ø.995,1.ØØ8)	Ø.Ø97

Programele cu mici modificări referitoare la dimensiunile tablourilor de lucru pot fi folosite pentru determinarea iteraţiei de start pentru sisteme neliniare în spaţii de dimensiune 6,7,... . Programul de calcul al gradului topologic necesită memorie de calculator mare, sau lucru cu fişiere ceea ce măreşte timpul de execuţie al programelor. Programele în versiunea actuală, pentru exemplul 2, au determinat iteraţia de start în aproximativ 30 de minute de calculator.

```
      PROGRAM POLIED
      PARAMETER(ND=5,MD=3Ø)
      DIMENSION P(Ø:MD,ND),A(ND+1,ND+1),IP(Ø:MD),IS(Ø:MD)
      COMMON/BO/N1
      COMMON/B1/A,D
      CALL ASSIGN(1,'SY:POLIED.DAT')
      CALL ASSIGN(2,'SY:POLIED.DAT')
      READ(1,*) N,M
      M=M-1
      N1=N+1
```

```
      IF(N1.GT.M) GOTO 151
      READ(1,101) ((P(I,J),J=1,N),I=Ø,M)
101   FORMAT((<N>F8.3))
      DO 1 I=Ø,N
      IP(I)=I
1     IS(I)=Ø
      DO 1Ø K=M,N,-1
      GOTO(2,3,4,5),N-1
2     DO 21 I1=Ø,K-2
      DO 21 I2=I1+1,K-1
      NZ=Ø
      NEG=Ø
      JP=Ø
      DO 22 I3=Ø,K
      IF(I3.EQ.I1.OR.I3.EQ.I2) GOTO 22
      DO 23 J=1,N
      A(1,J)=P(IP(I1),J)
      A(2,J)=P(IP(I2),J)
23    A(3,J)=P(IP(I3),J)
      DO 24 J=1,N1
24    A(J,N1)=1.
      CALL DET
      IF(D.NE.Ø.) GOTO 241
      NZ=NZ+1
      GOTO 22
241   IF(D.LT.Ø.) GOTO 25
      JP=JP+1
      IS(IP(I3))=1
      GOTO 22
25    NEG=NEG-1
      IS(IP(I3))=-1
```

Scanned with OKEN Scanner

```
22      CONTINUE
        IF(NZ.GT.Ø) GOTO 211
        IF(NEG.EQ.-1) GOTO 26
        IF(JP.EQ.1) GOTO 27
        GOTO 211
26      I=Ø
261     IF(IS(IP(I)).EQ.-1) GOTO 29
        I=I+1
        IF(I.LE.K) GOTO 261
        GOTO 153
27      I=Ø
271     IF(IS(IP(I)).EQ.1) GOTO 29
        I=I+1
        IF(I.LE.K) GOTO 271
        GOTO 153
29      WRITE(2,2Ø1) N,((L1),L1=Ø,N)
2Ø1     FORMAT(I2,',1,Ø.ØØØØ1,'/<N+1>(I2,','))
        WRITE(2,1Ø2)(P(IP(I1),J),J=1,N),(P(IP(I2),J),J=1,N),
        1(P(IP(I),J),J=1,N)
1Ø2     FORMAT((1X,<N>F8.3))
        IS(IP(I))=Ø
        IPI=IP(I)
        DO 291 J=I,K-1
291     IP(J)=IP(J+1)
        IP(K)=IPI
        DO 213 L1=Ø,M
213     IS(L1)=Ø
        GOTO 1Ø
211     DO 212 L1=Ø,M
212     IS(L1)=Ø
21      CONTINUE
```

Scanned with OKEN Scanner

```
      GOTO 154
3     DO 31 I1=Ø,K-3
      DO 31 I2=I1+1,K-2
      DO 31 I3=I2+1,K-1
      NZ=Ø
      NEG=Ø
      JP=Ø
      DO 32 I4=Ø,K
      IF(I4.EQ.I1,OR.I4.EQ.I2.OR.I4.EQ.I3) GOTO 32
      DO 33 J=1,N
      A(1,J)=P(IP(I1),J)
      A(2,J)=P(IP(I2),J)
      A(3,J)=P(IP(I3),J)
33    A(4,J)=P(IP(I4),J)
      DO 34 J=1,N1
34    A(J,N1)=1.
      CALL DET
      IF(D.NE.Ø.) GOTO 341
      NZ=NZ+1
      GOTO 32
341   IF(D.LT.Ø.) GOTO 35
      JP=JP+1
      IS(IP(I4))=1
      GOTO 32
35    NEG=NEG-1
      IS(IP(I4))=-1
32    CONTINUE
      IF(NZ.GT.Ø) GOTO 311
      IF(NEG.EQ.-1) GOTO 36
      IF(JP.EQ.1) GOTO 37
      GOTO 311
```

Scanned with OKEN Scanner

```
36      I=Ø
361     IF(IS(IP(I)).EQ.-1) GOTO 39
        I=I+1
        IF(I.LE.K) GOTO 361
        GOTO 153
37      I=Ø
371     IF(IS(IP(I)).EQ.1) GOTO 39
        I=I+1
        IF(I.LE.K) GOTO 371
        GOTO 153
39      WRITE(2,3Ø1) N,((L1),L1=Ø,N)
3Ø1     FORMAT(I2,',1,Ø.ØØØØ1,'/<N+1>(I2,','))
        WRITE(2,1Ø2) (P(IP(I1),J),J=1,N),(P(IP(I2),J),J=1,N),
        1(P(IP(I3),J),J=1,N),(P(IP(I),J),J=1,N)
        IS(IP(I))=Ø
        IPI=IP(I)
        DO 391 J=I,K-1
391     IP(J)=IP(J+1)
        IP(K)=IPI
        DO 313 L1=Ø,M
313     IS(L1)=Ø
        GOTO 1Ø
311     DO 312 L1=Ø,M
312     IS(L1)=Ø
31      CONTINUE
        GOTO 154
4       DO 41 I1=Ø,K-4
        DO 41 I2=I1+1,K-3
        DO 41 I3=I2+1,K-2
        DO 41 I4=I3+1,K-1
        NZ=Ø
```

Scanned with OKEN Scanner

```
      NEG=0
      JP=0
      DO 42 I5=0,K
      IF(I5.EQ.I1.OR.I5.EQ.I2.OR.I5.EQ.I3.OR.I5.EQ.I4) GOTO 42
      DO 43 J=1,N
      A(1,J)=P(IP(I1),J)
      A(2,J)=P(IP(I2),J)
      A(3,J)=P(IP(I3),J)
      A(4,J)=P(IP(I4),J)
43    A(5,J)=P(IP(I5),J)
      DO 44 J=1,N1
44    A(J,N1)=1.
      CALL DET
      IF(D.NE.0.) GOTO 441
      NZ=NZ+1
      GOTO 42
441   IF(D.LT.0.) GOTO 45
      JP=JP+1
      IS(IP(I5))=1
      GOTO 42
45    NEG=NEG-1
      IS(IP(I5))=-1
42    CONTINUE
      IF(NZ.GT.0) GOTO 411
      IF(NEG.EQ.-1) GOTO 46
      IF(JP.EQ.1) GOTO 47
      GOTO 411
46    I=0
461   IF(IS(IP(I)).EQ.-1) GOTO 49
      I=I+1
      IF(I.LE.K) GOTO 461
```

Scanned with OKEN Scanner

```
      GOTO 153
47    I=Ø
471   IF(IS(IP(I)).EQ.1) GOTO 49
      I=I+1
      IF(I.LE.K) GOTO 471
      GOTO 153
49    WRITE(2,4Ø1) N,((L1),L1=Ø,N)
4Ø1   FORMAT(I2,',1,Ø.ØØØØ1,'/<N+1>(I2,','))
      WRITE(2,1Ø2) (P(IP(I1),J),J=1,N),(P(IP(I2),J),J=1,N),
     1(P(IP(I3),J),J=1,N),(P(IP(I4),J),J=1,N),(P(IP(I),J),J=1,N)
      IS(IP(I))=Ø
      IPI=IP(I)
      DO 491 J=I,K-1
491   IP(J)=IP(J+1)
      IP(K)=IPI
      DO 413 L1=Ø,M
413   IS(L1)=Ø
      GOTO 1Ø
411   DO 412 L1=Ø,M
412   IS(L1)=Ø
41    CONTINUE
      GOTO 154
      DO 51 I1=Ø,K-5
      DO 51 I2=I1+1,K-4
      DO 51 I3=I2+1,K-3
      DO 51 I4=I3+1,K-2
      DO 51 I5=I4+1,K-1
      NZ=Ø
      NEG=Ø
      JP=Ø
      DO 52 I6=Ø,K
```

Scanned with OKEN Scanner

```
      IF(I6.EQ.I1.OR.I6.EQ.I2.OR.I6.EQ.I3.OR.I6.EQ.I4.OR.
     1 I6.EQ.I5) GOTO 52
      DO 53 J=1,N
      A(1,J)=P(IP(I1),J)
      A(2,J)=P(IP(I2),J)
      A(3,J)=P(IP(I3),J)
      A(4,J)=P(IP(I4),J)
      A(5,J)=P(IP(I5),J)
53    A(6,J)=P(IP(I6),J)
      DO 54 J=1,N1
54    A(J,N1)=1.
      CALL DET
      IF(D.NE.Ø.) GOTO 541
      NZ=NZ+1
      GOTO 52
541   IF(D.LT.Ø.) GOTO 55
      JP=JP+1
      IS(IP(I6))=1
      GOTO 52
55    NEG=NEG-1
      IS(IP(I6))=-1
52    CONTINUE
      IF(NZ.GT.Ø) GOTO 511
      IF(NEG.EQ.-1) GOTO 56
      IF(JP.EQ.1) GOTO 57
      GOTO 511
56    I=Ø
561   IF(IS(IP(I)).EQ.-1) GOTO 59
      I=I+1
      IF(I.LE.K) GOTO 561
      GOTO 153
```

Scanned with OKEN Scanner

```
57      I=Ø
571     IF(IS(IP(I)).EQ.1) GOTO 59
        I=I+1
        IF(I.LE.K) GOTO 571
        GOTO 153
59      WRITE(2,5Ø1) N,((L1),L1=Ø,N)
5Ø1     FORMAT(I2,',1,Ø.ØØØØ1,'/<N+1>(I2,','))
        WRITE(2,1Ø2) (P(IP(I1),J),J=1,N),(P(IP(I2),J),J=1,N),
        1(P(IP(I3),J),J=1,N),(P(IP(I4),J),J=1,N),(P(IP(I5),J)J=1,N),
        1(P(IP(I),J),J=1,N)
        IS(IP(I))=Ø
        IPI=IP(I)
        DO 591 J=1,K-1
591     IP(J)=IP(J+1)
        IP(K)=IPI
        DO 513 L1=Ø,M
513     IS(L1)=Ø
        GOTO 1Ø
511     DO 512 L1=Ø,M
512     IS(L1)=Ø
51      CONTINUE
        GOTO 154
1Ø      CONTINUE
        STOP
151     WRITE(2,1Ø3)
1Ø3     FORMAT(1X,'NR. VIRFURILOR < DIM.SPATIULUI')
        STOP
153     WRITE(2,1Ø5)
1Ø5)    FORMAT(1X,'EROARE DE MARCAJ')
        STOP
154     WRITE(2,1Ø6)
```

Scanned with OKEN Scanner

```
1Ø6     FORMAT(1X,'NU ESTE SIMPLEX DE DESCOMPUNERE;EROARE')
        STOP
        END
        SUBROUTINE DET
        PARAMETER(ND=5)
        DIMENSION A(ND+1,ND+1)
        COMMON/EØ/N1
        COMMON/E1/A,D
        E=0.1E-6
        D=1.
        DO 5 I=1,N1-1
        AC=ABS(A(I,I))
        IF(AC.GT.E) GOTO 4
        AM=AC
        DO 1 L=I+1,N1
        AC=ABS(A(L,I))
        IF(AM.GE.AC) GOTO 1
        IN=L
        AM=AC
1       CONTINUE
        IF(AM.GT.E) GOTO 2
        D=Ø.
        RETURN
2       D=-D
        DO 3 L=I,N1
        AC=A(I,L)
        A(I,L)=A(IN,L)
3       A(IN,L)=AC
4       AC=A(I,I)
        D=D*AC
        DO 5 K=I+1,N1
```

Scanned with OKEN Scanner

```
      AM=A(K,I)/AC
      DO 5 J=I,N1
5     A(K,J)=A(K,J)-A(I,J)*AM
      D=D*A(N1,N1)
      RETURN
      END

      PROGRAM STENG
      INTEGER*4 ID,IE
      INCLUDE 'STENG.FTN;1'
      COMMON/E0/N
      COMMON/E1/A(ND+1,ND+1),D
      COMMON/E2/V(ND),F(ND)
      DIMENSION IND(N1,0:ND)
      DIMENSION P(0:N2,ND)
      CALL ASSIGN(1,'SY:POLIED.LST')
      CALL ASSIGN(2,'SY:STENG.LST')
99    READ(1,1,END=100) N,M,EPS
1     FORMAT(2I6,F8.6)
      IF(N.GT.ND.OR.M.GT.N1) GOTO 28
      READ(1,2) ((IND(I,J),J=0,N),I=1,M)
2     FORMAT((<N+1>I4))
      LI=0
      DO 3 I=1,M
      DO 3 J=0,N
      L=IND(I,J)
      IF(LI.LT.L) LI=L
3     CONTINUE
      IF(LI.GT.N2) GOTO 28
      READ(1,4) ((P(I,J),J=1,N),I=0,LI)
4     FORMAT((1X,<N>F8.3))
```

Scanned with OKEN Scanner

```
      LII=LI
      X=1.
      K=Ø
      DO 1Ø L=1,M
      DO 7 I=1,N+1
7     A(I,1)=1.
      DO 8 I=1,N+1
      DO 8 J=1,N
8     A(I,J+1)=P(IND(L,I-1),J)
      N=N+1
      CALL DET
      N=N-1
      IF(D.NE.Ø) GOTO 1Ø
      WRITE(2,9) L
9     FORMAT(1X,'SIMPLEXUL ',I3,' NU ARE VECTORI LINIARI INDEPEN-
      DENTI')
      K=1
1Ø    CONTINUE
      IF(K.EQ.1) STOP
      IC=1
      DO 11 I=1,N
11    IC=IC*2*I
      FIC=FLOAT(IC)
      DD=1.7E38
12    DE=DD
      IS=Ø
      DO 17 L=1,M
      DO 13 I=1,N+1
13    A(I,1)=1.
      DO 14 I=1,N+1
      DO 14 J=1,N
```

Scanned with OKEN Scanner

```
14      A(I,J+1)=P(IND(L,I-1),J)
        N=N+1
        CALL DET
        N=N-1
        IT=IFIX(SIGN(X,D))
        DO 17 K=Ø,N
        DO 16 I=Ø,N
        IF(K.EQ.I) GOTO 16
        DO 15 J=1,N
15      V(J)=P(IND(L,I),J)
        CALL FUNCT
        J=1
151     IF(ABS(F(J)).GE.EPS) GOTO 153
        J=J+1
        IF(J.LE.N) GOTO 151
        WRITE(2,152)(V(J),J=1,N)
152     FORMAT(1X,'O SOLUTIE:',<N>F8.3/1X,'REDEFINITI POLIGONUL P')
        STOP
153     DO 16 J=1,N
        IF(I.LT.K) A(I+1,J)=SIGN(X,F(J))
        IF(I.GT.K) A(I,J)=SIGN(X,F(J))
16      CONTINUE
        CALL DET
        ID=NINT(D)
17      IS=IS+IT*ID*(-1)**K
        DD=FLOAT(IS)/FIC
        ID=NINT(DD)
        IF(ABS(DD-DE).LT.EPS.AND.ABS(DD-ID).LT.EPS) GOTO 26
        KM=Ø
        DO 24 L=1,M
        DM=Ø.
```

Scanned with OKEN Scanner

```
      DO 2Ø I=Ø,N-1
      DO 2Ø K=I+1,N
      DS=Ø.
      DO 19 J=1,N
19    DS=DS+(P(IND(L,I),J)-P(IND(L,K),J))**2
      IF(DS.LE.DM) GOTO 2Ø.
      DM=DS
      KI=I
      K2=K
2Ø    CONTINUE
      KM=KM+1
      DO 21 J=1,N
21    P(LI+KM,J)=(P(IND(L,K1),J)+P(IND(L,K2),J))/2.
      DO 22 J=Ø,K1-1
22    IND(M+L,J)=IND(L,J)
      IND(M+L,K1)=LI+KM
      DO 23 J=K1+1,N
23    IND(M+L,J)=IND(L,J)
      IND(L,K2)=LI+KM
24    CONTINUE
      M=M+KM
      LI=LI+KM
      IF(M.LE.N1.AND.LI.LE.N2) GOTO 12
      WRITE(2,25) M,LI
25    FORMAT(1X,'S-A DEPASIT SPATIUL ALOCAT '/
     11X,'NR.SIMPLEXE : ',I5,'NR.PUNCTE:',I5/)
26    WRITE(2,5) N,ID
5     FORMAT(1X,I1,',',I1,',␣')
      WRITE(2,6) ((P(I,J),J=1,N),I=Ø,LII)
6     FORMAT((1X,<N>F8.3))
      GOTO 99
```

Scanned with OKEN Scanner

```
27      FORMAT(1X,'M.= ',I5,' GRADUL TOPOLOGIC = ',I7)
100     STOP
28      WRITE(2,29) N,ND,M,N1,L,N2
29      FORMAT(1X,'N=',I4,' ND=',I4,' M=',I4,' N1=',I4,' L=',I4,
        1' N2=',I4)
        STOP
        END
        SUBROUTINE DET
        INCLUDE 'STENG.FTN;1'
        COMMON /E0/N
        COMMON /E1/A(ND+1,ND+1),D
        E=0.1E-6
        D=1.
        DO 5 I=1,N-1
        AC=ABS(A(I,I))
        IF(AC.GT.E) GOTO 4
        AM=AC
        DO 1 L=I+1,N
        AC=ABS(A(L,I))
        IF(AM.GE.AC) GOTO 1
        IN=L
        AM=AC
1       CONTINUE
        IF(AM.GT.E) GOTO 2
        D=0.
        RETURN
2       D=-D
        DO 3 L=I,N
        AC=A(I,L)
        A(I,L)=A(IN,L)
3       A(IN,L)=AC
```

Scanned with OKEN Scanner

```
4       AC=A(I,I)
        D=D*AC
        DO 5 K=I+1,N
        AM=A(K,I)/AC
        DO 5 J=I,N
5       A(K,J)=A(K,J)-A(I,J)*AM
        D=D*A(N,N)
        RETURN
        END

STENG.FTN;1
C       ND=2,N1=3072,N2=3072
C       ND=3,N1=2048,N2=2052
C       ND=4,N1=1024,N2=1050
C       ND=5,N1=1024,N2=1050
        PARAMETER (ND=3,N1=2048,N2=2052)

        PROGRAM JUMT
        DIMENSION P(6,5),G(6,5),C(5)
        PARAMETER(EPS=0.1)
        CALL ASSIGN(1,'SY:STENG.LST')
        CALL ASSIGN(2,'SY:JUMT.LST')
        CALL ASSIGN(3,'TI:')
        KK=0
1       READ(1,*,END=100) N,ID,J1
        IF(ID.EQ.1.OR.ID.EQ.-1) GOTO 3
        READ(1,2,END=101) ((G(I,J),J=1,N),I=1,N+1)
2       FORMAT((1X,<N>F8.3))
        GOTO 1
3       READ(1,2,END=101) ((P(I,J),J=1,N),I=1,N+1)
        KK=KK+1
        DO 10 J=1,N
```

Scanned with OKEN Scanner

```
      C(J)=Ø.
      DO 11 I=L,N+1
11    C(J)=C(J)+P(I,J)
1Ø    C(J)=C(J)/(N+1)
      EP=Ø.
      DO 12 I=1,N+1
      DE=Ø.
      DO 13 J=1,N
13    DE=DE+(C(J)-P(I,J))**2
      IF(DE.LE.EP) GOTO 12
      EP=DE
12    CONTINUE
      EP=SQRT(EP)
      IF(EP.GE.EPS) GOTO 14
      WRITE(2,141) (C(J),J=1,N),EP
141   FORMAT(1X,'PASUL INITIAL ESTE:'/1X,<N>F8.3/
      11X,'CU PRECIZIA:',F8.6/)
      GOTO 1
14    DM=Ø.
      DO 4 I=1,N
      DO 4 K=I+1,N+1
      DS=Ø.
      DO 5 J=1,N
5     DS=DS+(P(I,J)-P(K,J))**2
      IF(DS.LE.DM) GOTO 4
      DM=DS
      K1=I
      K2=K
4     CONTINUE
      DO 6 I=1,K1
      DO 6 J=1,N
```

Scanned with OKEN Scanner

```
6       G(I,J)=P(I,J)
        DO 7 J=1,N
        IF(J1.NE.1.AND.J1.NE.2) G(K1,J)=(P(K1,J)+P(K2,J))/2.
        IF(J1.EQ.1) G(K1,J)=(P(K1,J)+P(K2,J))*51./100.
        IF(J1.EQ.2) G(K1,J)=(P(K1,J)+P(K2,J))*49./100.
7       CONTINUE
        DO 8 I=K1+1,N+1
        DO 8 J=1,N
8       G(I,J)=P(I,J)
        DO 9 J=1,N
        IF(J1.NE.1.AND.J1.NE.2) P(K2,J)=(P(K1,J)+P(K2,J))/2.
        IF(J1.EQ.1) P(K2,J)=(P(K1,J)+P(K2,J))*51./100.
        IF(J1.EQ.2) P(K2,J)=(P(K1,J)+P(K2,J))*49./100.
9       CONTINUE
        WRITE(2,15) N,((L1),L1=0,N)
15      FORMAT(I2,',1,0\0000',1,'/<N+1>(I2,','))
        WRITE(2,102) ((P(I,J),J=1,N),I=1,N+1)
102     FORMAT((1X,<N>F8.3))
        WRITE(2,15) N,((L1),L1=0,N)
        WRITE(2,102) ((G(I,J),J=1,N),I=1,N+1)
        WRITE(3,16) (C(J),J=1,N),EP
16      FORMAT(1X,'C=',<N>F8.3,' EP=',F14.7)
        STOP
101     WRITE(3,201)
201     FORMAT(1X,'EROARE CITIRE EOF')
        STOP
100     IF(KK.EQ.1) STOP
        WRITE(3,202)
202     FORMAT(1X,'GRADUL TOPOLOGIC AL SIMPLEXILOR ESTE 0')
        STOP
        END
```

Scanned with OKEN Scanner

3.3. Inversarea matricilor celulare tridiagonale şi simetrice

In cele ce urmează ne propunem să găsim un algoritm de generare a inversei matricei $A_{m^2} \in \mathcal{M}_{m^2}(\mathbb{R})$:

$$A_{m^2}=\begin{bmatrix} B_m & -C_m & & 0_m \\ -C_m & B_m & \ddots & \\ & \ddots & \ddots & -C_m \\ 0_m & & -C_m & B_m \end{bmatrix} \quad B_m = \begin{bmatrix} a & b & & 0 \\ b & a & \ddots & \\ & \ddots & \ddots & b \\ 0 & & b & a \end{bmatrix} \quad C_m = \begin{bmatrix} c & e & & 0 \\ e & c & \ddots & \\ & \ddots & \ddots & e \\ 0 & & e & c \end{bmatrix} \tag{3.11}$$

unde B_m , C_m , $0_m \in \mathcal{M}_m(\mathbb{R})$, iar 0_m este matricea nulă de ordinul m. Matricea (3.11) îndeplineşte condiţiile de matrice celulară, cu celule matrici tridiagonale şi bisimetrice ($a_{ij} = a_{ji}$ şi $a_{m+1-i,m+1-j} = a_{ij}$ pentru $\forall\, i,j = \overline{1,m}$) care sînt aşezate tridiagonal. Bisimetria matricii A_{m^2} rezultă evident.

Pentru început să ne ocupăm de găsirea unui algoritm ce generează inversa matricei B_m (3.11). Fie ecuaţia cu diferenţe :

$$bp_{k+1} + ap_k + bp_{k-1} = 0 \qquad ab \neq 0 \tag{3.12}$$

Dacă $p_0 = b$ şi $p_1 = a$ atunci soluţiile ecuaţiei (3.12) sînt:

1. $p_k = b^2[(1+Q)^{k+1}-(1-Q)^{k+1}]/[aQ(-\operatorname{sgn}(ab)q)^{k+1}]$ (3.13)

cu $q = 2|b|/|a|$ şi $Q = \sqrt{1-q^2}$ pentru $\forall\, k = \overline{0,m}$, dacă $b \in (0,|a|/2)$;

2. $p_k = b(k+1)(-\operatorname{sgn}(ab))^{k+1}$ pentru $\forall\, k = \overline{0,m}$ (3.14)

dacă $|b| = |a|/2$;

3. $p_k = [a\cdot\sin(kq)-b\cdot\sin((k-1)q)](-\operatorname{sgn}(ab))^{k+1}/\sin(q)$ (3.15)

cu $q = \arccos(|a|/(2\cdot|b|))$ pentru $\forall\, k = \overline{0,m}$, dacă $|b|>|a|/2$.

Dacă $b = 0$ atunci evident că $B_m^{-1} = a^{-1}I_m$. Dacă $b \neq 0$ atunci B_m^{-1} este :

$$B_m^{-1}=\begin{pmatrix} r_{m-1} & r_{m-2} & r_{m-3} & \cdots & r_2 & r_1 & r_0 \\ r_{m-2} & r_{m-1}+r_{m-3} & r_{m-2}+r_{m-4} & \cdots & r_3+r_1 & r_2+r_0 & r_1 \\ r_{m-3} & r_{m-2}+r_{m-4} & r_{m-1}+r_{m-3}+r_{m-5} & \cdots & r_4+r_2+r_0 & r_3+r_1 & r_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ r_2 & r_3+r_1 & r_4+r_2+r_0 & \cdots & r_{m-1}+r_{m-3}+r_{m-5} & r_{m-2}+r_{m-4} & r_{m-3} \\ r_1 & r_2+r_0 & r_3+r_1 & \cdots & r_{m-2}+r_{m-4} & r_{m-1}+r_{m-3} & r_{m-2} \\ r_0 & r_1 & r_2 & \cdots & r_{m-3} & r_{m-2} & r_{m-1} \end{pmatrix} \quad (316)$$

unde $r_k = p_k/(bp_m)$ pentru orice $k = \overline{0,m-1}$.

Afirmaţia se confirmă direct prin verificarea identităţii $B_m B_m^{-1} = I_m$. Elementele matricei B_m^{-1} sînt :

$$b_{ij}^{-1} = \sum_{k=1}^{i} r_{m+i+1-2k-j} \quad \text{dacă} \quad i+j \leqslant m+1 \quad \text{şi} \quad i \leqslant j$$

$$b_{ij}^{-1} = \sum_{k=1}^{j} r_{m+j+1-2k-i} \quad \text{dacă} \quad i+j \leqslant m+1 \quad \text{şi} \quad i > j$$

$$b_{ij}^{-1} = \sum_{k=1}^{m+1-j} r_{m+i+1-2k-j} \quad \text{dacă} \quad i+j > m+1 \quad \text{şi} \quad i \leqslant j$$

$$b_{ij}^{-1} = \sum_{k=1}^{m+1-i} r_{m+j+1-2k-i} \quad \text{dacă} \quad i+j > m+1 \quad \text{şi} \quad i > j .$$

Dacă notăm cu $P_0 = C_m \in \mathcal{M}_m(\mathbb{R})$, $P_1 = B_m \in \mathcal{M}_m(\mathbb{R})$ şi $P_k = C_m^{-1} B_m P_{k-1} - P_{k-2} \in \mathcal{M}_m(\mathbb{R})$ pentru orice $k = \overline{2,m}$, atunci inversa matricei $A_m 2$ va fi matricea :

$$A_m^{-1}2=\begin{pmatrix} R_{m-1} & R_{m-2} & R_{m-3} & \cdots & R_2 & R_1 & R_0 \\ R_{m-2} & R_{m-1}+R_{m-3} & R_{m-2}+R_{m-4} & \cdots & R_3+R_1 & R_2+R_0 & R_1 \\ R_{m-3} & R_{m-2}+R_{m-4} & R_{m-1}+R_{m-3}+R_{m-5} & \cdots & R_4+R_2+R_0 & R_3+R_1 & R_2 \\ \vdots & \vdots & \vdots & & \vdots & \vdots & \vdots \\ R_2 & R_3+R_1 & R_4+R_2+R_0 & \cdots & R_{m-1}+R_{m-3}+R_{m-5} & R_{m-2}+R_{m-4} & R_{m-3} \\ R_1 & R_2+R_0 & R_3+R_1 & \cdots & R_{m-2}+R_{m-4} & R_{m-1}+R_{m-3} & R_{m-2} \\ R_0 & R_1 & R_2 & \cdots & -R_{m-3} & R_{m-2} & R_{m-1} \end{pmatrix}$$

unde $B_k = P_m^{-1}.C_m^{-1}.P_k \in \mathcal{M}_m(R)$ pentru orice $k = \overline{1,m-1}$. Afirmaţia de mai sus rezultă dintr-o verificare directă a identităţii : $A_{m^2}.A_{m^2}^{-1} = I_{m^2}$. De remarcat că din fiecare matrice B_k este suficient de memorat numai sfertul " $i+j \leq m+1$ şi $i \leq j$ " din motive de bisimetrie a fiecărei matrici. Inversarea matricei P_n se realizează printr-o factorizare Cholesky iar inversarea matricei triunghiulare prin factorizare, algoritm mult mai eficient ca orice inversare clasică de matrici.

Adresa elementului (i,j) în vectorul care reţine doar sfertul " $i+j \leq n+1$ şi $i \leq j$ " va fi dată de algoritmul :

- dacă $i+j \leq m+1$ şi $i \geq j$ atunci $adr(i,j)=(j-1)(m+1-j)+i$
- dacă $i+j > m+1$ şi $i \leq j$ atunci $adr(i,j)=(m-j).j+m+1-I$
- dacă $i+j > m+1$ şi $i > j$ atunci $adr(i,j)=(m-i).i+m+1-j$
- dacă $i+j \leq m+1$ şi $i \leq j$ atunci $adr(i,j)=(i-1)(m+1-i)+j$

De remarcat că numărul de operaţii în algoritmii de înmulţire se reduce aproape la 1/4, datorită observaţiei de mai sus. De asemenea înmulţirea lui P_{k-1} cu B_m are un număr mult redus de operaţii faţă de o înmulţire obişnuită de matrici.

Fie $ib = (i-1)/m+1$ $jb = (j-1)/m+1$ $ip = i-ib.m$ şi $jp = j-jb.m$ pentru orice $i,j = \overline{1,m}$, operaţiile efectuîndu-se în numere întregi. Acestea fiind precizate putem scrie formulele ce generează elementele matricii $A_{m^2}^{-1}$:

- dacă $ib+jb \leq m+1$ şi $ib \leq jb$: $a_{ij}^{-1} = \sum_{k=1}^{ib} R(m+ib+1-2k-jb, adr(ip,jp))$
- dacă $ib+jb \leq m+1$ şi $ib > jb$: $a_{ij}^{-1} = \sum_{k=1}^{jb} R(m+jb+1-2k-ib, adr(ip,jp))$
- dacă $ib+jb > m+1$ şi $ib \leq jb$: $a_{ij}^{-1} = \sum_{k=1}^{m+1-jb} R(m+ib+1-2k-jb, adr(ip,jp))$
- dacă $ib+jb > m+1$ şi $ib > jb$: $a_{ij}^{-1} = \sum_{k=1}^{m+1-ib} R(m+jb+1-2k-ib, adr(ip,jp))$

unde a_{ij}^{-1} sînt elementele matricei $A_{m^2}^{-1}$.

Programul BB1 este un program de generare a inversei matricei tridiagonale B_m (3.11) și testarea corectitudinii algoritmului prin înmulțirea cu B_m.

```
      PROGRAM BB1
C
C     PROGRAM DE GENERARE A INVERSEI MATRICEI TRIDIAGONALE
C
C     a b Ø ... Ø
C     b a b ... Ø
C     . . .     .
C     . . .     .
C     . . .     .
C     Ø Ø Ø ... a
C
      PARAMETER (N=1Ø)
      DIMENSION R(Ø:N),BB(N)
      COMMON/B1/R/ER/A,B,M,A2,B2
      CALL ASSIGN(1,'SY:BB1.DAT')
      CALL ASSIGN(2,'SY:BB1.LST')
      READ(1,*) A2,B2,M
      IF(A2.EQ.Ø..OR.M.GT.N) GOTO 6
      IF(B2.EQ.Ø.) GOTO 11
      A=ABS(A2)
      B=ABS(B2)
      DO 1 I=Ø,M
1     R(I)=R(I)
11    DO 4 I=1,M
      DO 3 J=1,M
      BB(J)=Ø.
      DO 2 K=1,M
2     BB(J)=BB(J)+B1(I,K)*C(K,J)
```

```
3     CONTINUE
4     WRITE(2,5) (BB(K),K=1,M)
5     FORMAT(1X,<M>F8.5)
      STOP
6     WRITE(2,*) 'EROARE a SAU b SINT ZERO SAU M>N'
      STOP
      END
      FUNCTION P(I)
      COMMON/E2/A,B,M,A2,B2
      K=MOD(I+1,2)
      S=1
      IF(K.EQ.1) S=-1
      BB=2.*B
      IF(A.EQ.BB) THEN
        P=B*(I+1)
        GOTO 1
      ENDIF
      IF(A.LT.BB) THEN
        T=ACOS(A/BB)
        P=(A*SIN(I*T)-B*SIN((I-1)*T))/SIN(T)
        GOTO 1
      ENDIF
      T=BB/A
      T1=SQRT(1.-T*T)
      P=((1.+T1)**(I+1)-(1.-T1)**(I+1))/T**(I+1)
      P=P*B**B/(A*T1)
1     IF(A2*B2.GT.0.) P=S*P
      RETURN
      END
      FUNCTION B1(I,J)
```

Scanned with OKEN Scanner

```
      PARAMETER (N=1Ø)
      DIMENSION R(Ø:N)
      COMMON/E1/R/E2/A,D,M,A2,B2
      IF(B2.EQ.Ø.) THEN
        B1=Ø.
        IF(I.EQ.J) B1=1./A2
        RETURN
      ENDIF
      IF(I+J.LE.M+1.AND.I.LE.J) THEN
        B1=Ø.
        DO 1 K=1,I
1       B1=B1+R(M+I+1-2*K-J)
        GOTO 5
      ENDIF
      IF(I+J.LE.M+1.AND.I.GT.J) THEN
        B1=Ø.
        DO 2 K=1,J
2       B1=B1+R(M+J+1-2*K-I)
        GOTO 5
      ENDIF
      IF(I+J.GT.M+1.AND.I.LE.J) THEN
        B1=Ø.
        DO 3 K=1,M+1-J
3       B1=B1+R(M+I+1-2*K-J)
        GOTO 5
      ENDIF
      B1=Ø.
      DO 4 K=1,M+1-I
4     B1=B1+R(M+J+1-2*K-I)
5     B1=B1/(B*R(M))
      IF(B2.GT.Ø.) B1=-1.*B1
```

Scanned with OKEN Scanner

```
      RETURN
      END
      FUNCTION C(I,J)
      COMMON/B2/A,B,M,A2,B2
      IF(I.EQ.J) THEN
        C=A2
        RETURN
      ENDIF
      IF(IABS(I-J).EQ.1) THEN
        C=B2
        RETURN
      ENDIF
      C=Ø.
      RETURN
      END
```

Programul B5 este un program de generare a vectorilor $R_0, R_1, \ldots, R_m$ necesari pentru a obține inversa matricei Am^2. Acești vectori sînt memorați într-un fișier secvențial pe disc.

```
      PROGRAM B5
C          INVERSAREA MATRICEI A
C          B -C 0 ... 0
C         -C  B-C ... 0
C          0 -C B ... 0
C       A= .  . .    .
C          .  . .    .
C          .  . .    .
C          0  0 0 ... B
C          UNDE B SI C SINT MATRICI LA RINDUL LOR
C
```

```
C          a b Ø ... Ø
C          b a b ... Ø
C          Ø b a ... Ø
C       B= . . .     .
C          . . .     .
C          . . .     .
C          0 0 0 ... e
C
C          e f 0 ... 0
C          f e f ... 0
C          0 f e ... 0
C       C= . . .     .
C          . . .     .
C          . . .     .
C          0 0 0 ... e
C
C          SINT CINCI PARAMETER
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (M=5,M1=(M+1)/2,M2=(M+1-M1)*M1)    !1
      DIMENSION P(Ø:M,M2),C(Ø:M2),D(M2)
      COMMON/B2/A,B
      CALL ASSIGN(1,'DD:B5.DAT')
      CALL ASSIGN(2,'TI:')
      CALL ASSIGN(3,'DD:B5.MAN')
      READ(1,*) A,B,E,F
      IF(A*E.EQ.Ø) THEN
       WRITE(2,*)'EROARE a sau e este = cu Ø.STOP'
        STOP
      ENDIF
      K=(M+1)/2
C              GENERAREA MATRICSI P(Ø,*)=I
```

Scanned with OKEN Scanner

```
      DO 1 I=1,K
      I1=M+1-I
      DO 1 J=I,I1
      I2=L(I,J,M)
      P(Ø,I2)=Ø.
      IF(I.EQ.J) P(Ø,I2)=E
      IF(IABS(I-J).EQ.1) P(Ø,I2)=F
1     CONTINUE
C
C           INVERSAREA MATRICEI I
      CALL BB1(E,F)
C
C           GENERAREA MATRICEI P(1,=)=B
      DO 2 I=1,K
      I1=M+1-I
      DO 2 J=I,I1
      I2=L(I,J,M)
      P(1,I2)=Ø.
      IF(I.EQ.J) P(1,I2)=A
      IF(IABS(I-J).EQ.1) P(1,I2)=B
2     CONTINUE
      DO 4 J=2,M-1
      DO 3 I=1,M2
3     C(I)=P(J-1,I)
      CALL CORIB(C)
      DO 4 I=1,M2
4     P(J,I)=C(I)-P(J-2,I)
      DO 5 I=1,M2
5     C(I)=P(M-1,I)
      CALL CORIB(C)
      DO 6 I=1,M2
```

Scanned with OKEN Scanner

```
6     C(I)=C(I)-P(M-2,I)
      CALL INVERS(C)
      DO 8 N=Ø,M-1
      DO 7 I=1,K
      DO 7 J=I,M+1-I
      I1=L(I,J,M)
      D(I1)=Ø.
      DO 7 K1=1,M
7     D(I1)=D(I1)+C(L(I,K1,M))*P(N,L(K1,J,M))
      DO 8 I=1,M2
8     P(N,I)=D(I)
      WRITE(3) M
      WRITE(3) ((P(I,J),J=1,M2),I=Ø,M-1)
      STOP
      END
      FUNCTION L(I,J,M)
      IF(J.EQ.Ø.OR.J.EQ.M+1) THEN
        L=Ø
        RETURN
      ENDIF
      IF(I+j.LE.M+1.AND.I.GT.J) THEN
        J1=I
        I1=J
        GOTO 1
      ENDIF
      IF(I+J.GT.M+1.AND.I.LE.J) THEN
        J1=M+1-I
        I1=M+1-J
        GOTO 1
      ENDIF
      IF(I+J.GT.M+1.AND.I.GT.J) THEN
```

Scanned with OKEN Scanner

```
      I1=M+1-I
      J1=M+1-J
      GOTO 1
    ENDIF
    I1=I
    J1=J
1   L=(I1-1)*(M+1-I1)+J1
    RETURN
    END
    FUNCTION LIN(I,M)
    LIN=(I-1)*(2*M-I)/2
    RETURN
    END
    SUBROUTINE CORIB(C)
    IMPLICIT REAL*8 (A-H,O-Z)
    PARAMETER (M=5,M1=(M+1)/2,M2=(M+1-M1)*M1)  !2
    DIMENSION C(0:M2),D(M2)
    COMMON/B2/A,B
    C(0)=0.
    K=(M+1)/2
    DO 1 I=1,K
    I1=M+1-I
    DO 1 J=I,I1
    I2=L(I,J,M)
1   D(I2)=B*C(L(I,J-1,M))+A*C(I2)+B*C(L(I,J+1,M))
    DO 2 I=1,K
    I1=M+1-I
    DO 2 J=I,I1
    I2=L(I,J,M)
    C(I2)=0.
    DO 2 N=1,M
```

Scanned with OKEN Scanner

```
2     C(I2)=C(I2)+B1(I,N)*D(L(N,J,M))
      RETURN
      END
      SUBROUTINE BB1(A2,B2)
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (M=5)                    ! 3
      DIMENSION R(0:M)
      COMMON/B1/R,A,B
      A=A2
      B=B2
      IF(B2.EQ.0.) RETURN
      DO 1 I=0,M
1     R(I)=P(I,A2,B2)
      RETURN
      END
      REAL*8 FUNCTION P(I,A2,B2)
      IMPLICIT REAL*8 (A-H,O-Z)
      A=ABS(A2)
      B=ABS(B2)
      K=MOD(I+1,2)
      S=1.
      IF(K.EQ.1) S=-1.
      BB=2.*B
      IF(A.EQ.BB) THEN
        P=B*(I+1)
        GOTO 1
      ENDIF
      IF(A.LT.BB) THEN
        T=ACOS(A/BB)
        P=(A*SIN(I*T)-B*SIN((I-1)*T))/SIN(T)
        GOTO 1
```

Scanned with OKEN Scanner

```
      ENDIF
      T=BB/A
      T1=SQRT(1.-T*T)
      P=((1.+T1)**(I+1)-(1.-T1)**(I+1))/T**(I+1)
      P=P*B**B/(A*T1)
1     IF(A2*B2.GT.Ø.) P=S*P
      RETURN
      END
      REAL*8 FUNCTIØN B1(I,J)
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (M=5)                !4
      DIMENSION R(Ø:M)
      COMMON/B1/R,A2,B2
      IF(B2.EQ.Ø.) THEN
        B1=Ø.
        IF(I.EQ.J) B1=1./A2
        RETURN
      ENDIF
      B=ABS(B2)
      IF(I+J.LE.M+1.AND.I.LE.J) THEN
        B1=Ø.
        DO 1 K=1,I
1       B1=B1+R(M+I+1-2*K-J)
        GOTO 5
      ENDIF
      IF(I+J.GT.M+1.AND.I.LE.J) THEN
        B1=Ø.
        DO 3 K=1,M+1-J
3       B1=B1+R(M+I+1-2*K-J)
        GOTO 5
```

Scanned with OKEN Scanner

```
      ENDIF
      B1=Ø.
      DO 4 K=1,M+1-I
4     B1=B1+R(M+J+1-2*K-I)
5     B1=B1/(B*R(M))
      IF(B2.GT.Ø.) B1=-1.*B1
      RETURN
      END
      SUBROUTINE INVERS(C)
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (M=5,M1=(M+1)/2,N=(M+1)*M/2,M2=(M+1-M1)*M1)
      DIMENSION U(N),C(Ø:M2)
      U(1)=SQRT(C(1))
      DO 1 J=2,M
1     U(J)=C(L(1,J,M))/U(1)
      DO 4 K=2,M
      LK=LIN(K,M)
      Y=C(L(K,K,M))
      DO 2 I=1,K-1
      LI=LIN(I,M)
      X=U(LI+K)
2     Y=Y-X*X
      Y=SQRT(Y)
      U(LK+K)=Y
      DO 4 J=K+1,M
      X=C(L(K,J,M))
      DO 3 I=1,K-1
      LI=LIN(I,M)
3     X=X-U(LI+K)*U(LI+J)
4     U(LK+J)=X/Y
      U(1)=1./U(1)
```

Scanned with OKEN Scanner

```
      DO 5 I=2,M
5     U(I)=U(I)*U(1)
      DO 9 K=2,M
      LK=LIN(K,M)
      DO 6 I=1,K-1
      LI=LIN(I,M)
      DO 6 J=K+1,M
6     U(LI+J)=U(LI+J)-U(LI+K)*U(LK+J)/U(LK+K)
      DO 7 J=K+1,M
7     U(LK+J)=U(LK+J)/U(LK+K)
      DO 8 J=1,K-1
      LJ=LIN(J,M)
8     U(LJ+K)=-U(LJ+K)/U(LK+K)
9     U(LK+K)=1./U(LK+K)
      K=(M+1)/2
      DO 10 I=1,K
      LI=LIN(I,M)
      DO 10 J=I,M+1-I
      LJ=LIN(J,M)
      I2=L(I,J,M)
      C(I2)=0.
      DO 10 I1=J,M
10    C(I2)=C(I2)+U(LI+I1)*U(LJ+I1)
      RETURN
      END
```

Programul B5 citeşte fişierul secvenţial creat de programul B3 şi pe baza algoritmului enunţat mai sus generează elementele matricei $A_{m^2}^{-1}$. Totodată programul verifică corectitudinea algoritmilor şi a programului B3 prin verificarea identităţii $A_{m^2} \cdot A_{m^2}^{-1} = I_{m^2}$.

```
      PROGRAM B5
      IMPLICIT REAL*8 (A-H,O-Z)
C               SINT 5 PARAMETER
      PARAMETER (M=5,M1=(M+1)/2,M2=(M+1-M1)*M1)    ! 1
      DIMENSION P(0:M-1,M2),W(M*M)
      COMMON/B1/P
      CALL ASSIGN(1,'DD:B5.MAN')
      CALL ASSIGN(2,'DD:B5.LST')
      READ(1) N
      IF(M.NE.N) THEN
        WRITE(2,*) 'STOP. M#N'
        STOP
      ENDIF
      READ(1) ((P(I,J),J=1,M2),I=0,M-1)
      DO 2 I=1,M*M
      DO 1 J=1,M*M
      W(J)=0.0D+0
      DO 1 K=1,M*M
    1 W(J)=W(J)+A(I,K)*A1(K,J)
      WRITE(2,*) 'LINIA',I
      DO 11 K=1,M*M
      IF(I.NE.K) THEN
        IF(DABS(W(K)).LE.0.1D-14) THEN
          GOTO 11
        ELSE
          WRITE(2,*)'I=',I,' K=',K,' W=',W(K)
        ENDIF
      ELSE
        IF(DABS(W(K)-1.0D+0).LE.0.1D-14) THEN
          GOTO 11
        ELSE
```

Scanned with OKEN Scanner

```
      WRITE(2,*)'I=',I,' K=',K,' W=',W(K)
     ENDIF
    ENDIF
11  CONTINUE
2   CONTINUE
    STOP
    END
    REAL*8 FUNCTION A1(I,J)
    IMPLICIT REAL*8 (A-H,O-Z)
    PARAMETER (M=5,M1=(M+1)/2,M2=(M+1-M1)*M1)  ! 2
    DIMENSION P(0:M-1,M2)
    COMMON/B1/P
    IB=(I-1)/M
    JB=(J-1)/M
    IE=I-IB*M
    JE=J-JB*M
    IB=IB+1
    JB=JB+1
    KE=L(IE,JE,M)
    IF(IB+JB.LE.M+1.AND.IB.LE.JB) THEN
     A1=0.0D+0
     DO 1 K=1,IB
1    A1=A1+P(M+IB+1-2*K-JB,KE)
     RETURN
    ENDIF
    IF(IB+JB.LE.M+1.AND.IB.GT.JB) THEN
     A1=0.0D+0
     DO 2 K=1,JB
2    A1=A1+P(M+JB+1-2*K-IB,KE)
     RETURN
    ENDIF
```

Scanned with OKEN Scanner

```
      IF(IB+JB.GT.M+1.AND.IB.LE.JB) THEN
        A1=Ø.ØD+Ø
        DO 3 K=1,M+1-JB
3       A1=A1+P(M+IB+1-2*K-JB,KE)
        RETURN
      ENDIF
      A1=Ø.ØD+Ø
      DO 4 K=1,M+1-IB
4     A1=A1+P(M+JB+1-2*K-IB,KE)
      RETURN
      END
      FUNCTION L(I,J,M)
      IF(J.EQ.Ø.OR.J.EQ.M+1) THEN
        L=Ø
        RETURN
      ENDIF
      IF(I+1.LE.M+1.AND.I.GT.J) THEN
        J1=I
        I1=J
        GOTO 1
      ENDIF
      IF(I+J.GT.M+1.AND.I.LE.J) THEN
        J1=M+1-I
        I1=M+1-J
        GOTO 1
      ENDIF
      IF(I+J.GT.M+1.AND.I.GT.J) THEN
        I1=M+1-I
        J1=M+1-J
        GOTO 1
      ENDIF
```

Scanned with OKEN Scanner

```
      I1=I
      J1=J
1     L=(I1-1)*(M+1-I1)+J1
      RETURN
      END
      REAL*8 FUNCTION A(I,J)
      IMPLICIT REAL*8 (A-H,O-Z)
      PARAMETER (M=5)       ! 3
      IF(I.EQ.J) THEN
        A=4.ØD+Ø
        RETURN
      ENDIF
      I1=(I-1)/M
      J1=(J-1)/M
      IF(IABS(I-J).EQ.1.AND.I1.EQ.J1) THEN
        A=-1.ØD+Ø
        RETURN
      ENDIF
      IF(IABS(I-J).EQ.M.AND.IABS(I1-J1).EQ.1) THEN
        A=-1.ØD+Ø
        RETURN
      ENDIF
      A=Ø.ØD+Ø
      RETURN
      END
```

3.4. Metoda iteraţiilor simple pentru ecuaţii de forma (2.12)

În acest paragraf se studiază convergenţa locală a unei variante a metodei iteraţiilor simple pentru ecuaţii cu diferenţe corespunzătoare problemelor la limită bilocale neliniare.

Ca aplicaţie se obţine o metodă iterativă pentru ecuaţiile cu diferenţe corespunzătoare unei probleme la limită bilocală slab neliniară.

3.4.1. Algoritmul de calcul şi analiza convergenţei. In [5(1)] este studiată metoda iteraţiilor simple $Ax^{k+1} = \phi x^k$ pentru rezolvarea numerică a ecuaţiilor cu diferenţe de forma (2.4) corespunzătoare unei probleme la limită bilocală slab neliniară de forma (2.2). Această metodă generează un şir convergent $\{x^k\}$ dacă funcţia f este Lipschitz-continuă şi dacă constanta Lipschitz L satisface relaţia $L < 8$. Să observăm că fiecare pas de iteraţie comportă rezolvarea unui sistem liniar cu matricea A, cu coloana termenilor liberi $\bar{\phi} x^k$, ceea ce reprezintă un volum substanţial de calcule. In cazuri particulare, de exemplu dacă A este lacunară pentru orice $x \in B^n$, diagonală sau bloc diagonală, se pot utiliza algoritmi speciali de stocare în memoria internă şi de prelucrare (vezi paragraful 4.3) [15(6)], [32(2)], [58(1)] şi volumul de calcul devine rezonabil.

Fie B un spaţiu Banach $A: B \longrightarrow L(B)$ şi $\bar{\phi} : B \longrightarrow B$, operatori neliniari. Vom considera ecuaţia

$$A(x)x - \bar{\phi}x = 0 \qquad (3.17)$$

şi următoarea metodă iterativă [59(1)] [58(2)]

$$A(x^k)x^{k+1} - \bar{\phi}x^k = 0 \ , \ x^0 \text{ arbitrar.} \qquad (3.18)$$

pe care o vom numi metoda iteraţiilor simple. Să observăm că această metodă generalizează metoda amintită şi că comportă de asemenea rezolvarea unei ecuaţii liniare la fiecare pas.

Teorema 3.8. Presupunem că operatorii A şi $\bar{\phi}$ sînt Lipschitz continui pe B (fie L_A şi respectiv $L_{\bar{\phi}}$ constantele Lipschitz) şi că operatorul $A(x)$ are invers mărginit pentru orice x aparţinînd lui B. Presupunem de asemenea că $\| A(x)^{-1} \| \leq \alpha$, $\| \bar{\phi} x \| \leq \beta$ pentru orice x aparţinînd lui B şi că con-

stantele L_A , L_Φ, α, β satisfac relaţia

$$\alpha L_\Phi + \alpha^2 \beta L_A < 1 \qquad (3.19)$$

atunci ecuaţia (3.17) are o unică soluţie x^* aparţinînd lui B iar iteraţia (3.18) generează un şir $\{x^k\}$ convergent la x^*.

Demonstraţie.

Fie $T: B \longrightarrow B$ definit de $Tx = A(x)^{-1}\bar{\Phi}(x)$, evident, T este definit pentru orice x aparţinînd lui B iar iteraţia (3.18) este identică cu $x^{k+1} = Tx^k$. Avem

$$\|Tx - Ty\| \leq \|A(x)^{-1}\bar{\Phi}(x) - A(x)^{-1}\bar{\Phi}(y)\| + \|A(x)^{-1}\bar{\Phi}(y) - A(y)^{-1}\bar{\Phi}(y)\|$$
$$\leq \|A(x)^{-1}\| \|\bar{\Phi}(x) - \bar{\Phi}(y)\| + \|\bar{\Phi}(y)\| \|A(x)^{-1} - A(y)^{-1}\| \leq \alpha L_\Phi \|x-y\| +$$
$$+ \beta \|A(x)^{-1}\| \|[A(x) - A(y)]A(y)^{-1}\| \leq (\alpha L_\Phi + \alpha^2 \beta L_A)\|x-y\|$$

prin urmare T este o contracţie şi teorema rezultă din principiul contracţiei al lui Banach.

Observaţia 3.3. Un rezultat similar se poate obţine şi dacă condiţiile din teoremă au loc numai pe o anumită mulţime D din B. In acest caz este necesar să se impună anumite condiţii suplimentare, de exemplu ca $T(D)$ să fie inclusă în D sau ca ecuaţia (3.17) să aibe o soluţie. Teorema care urmează este în esenţă o variantă a teoremei (3.8).

Teorema 3.9. Presupunem că operatorii A şi $\bar{\Phi}$ sînt diferenţiabili Fréchet pe B şi că $A(x)$ are invers mărginit pentru orice x aparţinînd lui B. Dacă în plus $\|A(x)^{-1}\| \leq \alpha$ şi

$$\|[A(x)^{-1}\bar{\Phi}x]'_x + A(x)^{-1}\bar{\Phi}'(x)\| \leq \beta < 1 \qquad (3.20)$$

pentru orice x aparţinînd lui B, atunci ecuaţia (3.17) are o soluţie unică $x^* \in B$ iar iteraţia (3.18) generează un şir $\{x^k\}$ convergent la x^*.

Demonstraţie. Vom aplica formula (3.1) pentru a calcula derivata operatorului T definit ca mai sus. Să notăm cu A^{-1} ope-

ratorul definit de $A^{-1}x = A(x)^{-1}$ şi să arătăm că A^{-1} este diferenţiabil Fréchet pe B. Fie $K: B \longrightarrow L(B)$ un operator definit pentru orice x^0 aparţinînd lui B de

$$Kh = A(x^0)^{-1}A'(x^0)h\,A(x^0)^{-1}.$$

Se vede că operatorul K este liniar şi mărginit. Avem :

$$\| A^{-1}x - A^{-1}x^0 - K(x-x^0)\| \le \|A(x)^{-1} - A(x^0)^{-1} + A(x)^{-1}A'(x^0)(x-x^0)A(x^0)^{-1}\| +$$
$$\|\left[A(x)^{-1} - A(x^0)^{-1}\right]A'(x^0)(x-x^0)A(x^0)^{-1}\| \le$$
$$\le \| A(x)^{-1}\left[A(x) - A(x^0) - A'(x^0)(x-x^0)\right]A(x^0)^{-1}\| +$$
$$+ \|A(x)^{-1} - A(x^0)^{-1}\|\,\|A'(x^0)\|\,\|x-x^0\|\,\|A(x^0)^{-1}\| \le$$
$$\le (\alpha^2\varepsilon + \alpha\|A'(x^0)\|\,\|A(x)^{-1} - A(x^0)^{-1}\|)\,\|x-x^0\| .$$

Deoarece $A(x)^{-1} \longrightarrow A(x^0)^{-1}$ dacă $x \longrightarrow x^0$ rezultă că operatorul K este derivata Fréchet a lui A^{-1}. Acum conform formulei de derivare (3.1) rezultă

$$T'(x) = \left[A(z)^{-1}\bar{\Phi}\,x\right]'_x + A(x)^{-1}\bar{\Phi}'(x)$$

astfel încît $\|T'(x)\| \le \beta < 1$ pentru orice x aparţinînd lui B şi ţinînd cont de formula de medie avem :

$$\|Ty - Tx\| \le \sup_{t\in[0,1]} T'(x + t(y-x))\,\|y-x\| \le \beta\,\|y-x\| .$$

T este o contracţie şi teorema 3.9 rezultă din principiul contracţiei.

Observaţia 3.4. Teorema (3.9) nu este un corolar al teoremei 3.8. De exemplu, dacă $B = R$ iar $A(x) = 2$, $\bar{\Phi}(x) = x$ atunci toate condiţiile teoremei 3.9 sînt îndeplinite dar $\bar{\Phi}$ nu este o funcţie mărginită. De asemenea, condiţia (3.2o) nu implică mărginirea derivatelor Fréchet a operatorilor A şi $\bar{\Phi}$ din care să rezulte continuitatea Lipschitz a acestora.

3.4.2. Aplicaţie la rezolvarea ecuaţiilor cu diferenţe corespunzătoare problemelor la limită bilocale cvasiliniare.

Considerăm problema la limită (2.11) şi ecuaţiile cu dife-

rențe corespunzătoare (2.12). Vom scrie aceste ecuații sub o formă puțin modificată și anume vom pune $A(x) = A_1 + h^2 A_2(x)$, unde $A_1 = -h^2 A$ și $A_2(x) = \mathrm{Diag}[a(x_1),\dots,a(x_n)]$, A fiind matricea (2.1). De asemenea vom nota $\Phi x = h^2[f(t_1,x_1),\dots, \dots, f(t_n,x_n)]^T$. Obținem astfel o ecuație neliniară de forma (3.17) pe un spațiu finit dimensional pentru rezolvarea căreia vom aplica iterația (3.18). Vom utiliza în $\mathbb{R}^n$ norma $\|\cdot\|$ definită de $\|x\| = \max_{i=\overline{1,n}} |x_i|$, unde x_i sînt componentele lui x iar în spațiul $L(\mathbb{R}^n)$ norma operatorială corespunzătoare. Vom face ipoteza că funcțiile a și f satisfac următoarele condiții :

$$\begin{aligned} &|a(u)| \le M_a \qquad \forall\, u \in \mathbb{R} \\ &|a(u)-a(v)| \le L_a|u-v| \qquad \forall\, u,v \in \mathbb{R} \\ &|f(t,u)| \le M_f \qquad \forall\, t,u \in \mathbb{R} \\ &|f(t,u)-f(t,v)| \le L_f|u-v| \qquad \forall\, t,u,v \in \mathbb{R} \end{aligned} \tag{3.21}$$

și vom presupune că constantele M_a , L_a , M_f , L_f satisfac relațiile

$$\begin{aligned} &M_a < 8 \\ &\frac{L_f}{8-M_a} + \frac{L_a M_f}{(8-M_a)^2} < 1 \end{aligned} \tag{3.22}$$

In aceste ipoteze vom arăta că toate condițiile teoremei (3.8) sînt satisfăcute. Pentru simplicitate vom presupune că n este un număr par.

Dacă notăm cu a_{ij}^{-1} elementul (i,j) al matricei A_1^{-1} atunci $\big([13(5)]\,[(32(2)]\,[58(1)]\big)$ conform (3.14) și (3.16) unde $b = 1$ și $a = -2$ avem

$$a_{ij}^{-1} = \begin{cases} \dfrac{i(n-j+1)}{n+1} & i \le j \\ \dfrac{j(n-i+1)}{n+1} & i \ge j \end{cases}$$

Atunci $\|A_1^{-1}\| = \max_{i=\overline{1,n}} \sum_{j=1}^{n} |a_{ij}^{-1}| = \max_{i=\overline{1,n}} \dfrac{i(n+1-i)}{2}$, maxim care se

atinge pentru $i = \left[\frac{n+1}{2}\right]$, deci $\|A_1^{-1}\| = \frac{(n+1)^2}{8}$. Pe de altă parte, norma matricei $h^2A_2(x)$ este mărginită de h^2M_a şi prin urmare dacă $M_a < 8$ atunci matricea $A(x) = A_1 + h^2A_2(x)$ este inversabilă şi $|A(x)^{-1}| \leq (n+1)^2/(8-M_a)$. Deci constanta α din teorema (3.8) este $\alpha = (n+1)^2/(8-M_a)$. Ţinînd cont de (3.21) avem

$$\|A(x)-A(y)\| = h^2 \max_{i=\overline{1,n}} |a(x_i)-a(y_i)| \leq h^2L_a\|x-y\|$$

$$\|\tilde{\phi}x-\tilde{\phi}y\| = h^2 \max_{i=\overline{1,n}} |f(t_i,x_i)-f(t_i,y_i)| \leq h^2L_f\|x-y\|$$

$$\|\phi x\| = h^2 \max_{i=\overline{1,n}} |f(t_i,x_i)| \leq h^2M_f .$$

Prin urmare, celelalte constante din teorema (3.8) sînt $L_A = h^2L_a$, $L_{\tilde{\phi}} = h^2L_f$, $\beta = h^2M_f$. Inegalitatea (3.19) se reduce la inegalitatea (3.22). In acest mod, toate condiţiile teoremei (3.8) sînt satisfăcute, ecuaţia (2.11) are o soluţie unică x^* iar şirul $\{x^k\}$ generat de (3.18) converge la x^*.

Observaţia 3.5. Dacă $a(u) = o$ pentru orice $u \in \mathbb{R}$, atunci $M_a = L_a = 0$ iar (3.22) se reduce la condiţia cunoscută $L_f < 8$ [5(1)].

Observaţia 3.6. Se vede uşor că condiţia (3.19) este independentă de n. Aceasta înseamnă că numărul de iteraţii depinde numai de precizia cu care dorim să obţinem soluţia x^*. In conformitate cu testele numerice efectuate (a se vedea paragraful 4.1) numărul de iteraţii se încadrează în limitele evaluate după relaţia (3.19), rămînînd constant cînd n creşte.

Observaţia 3.7. In cazuri particulare, condiţiile de convergenţă ale iteraţiei (3.18) pot să fie îndeplinite în timp ce condiţiile de convergenţă ale iteraţiei simple ($Ax^{k+1} = x^k$) nu sînt satisfăcute. De exemplu în cazul ecuaţiilor cu diferenţe corespunzătoare problemei la limită $u''-u \sin(u) + 1 = o$ $u(o) = u(1) = o$, condiţiile de convergenţă ale iteraţiei (3.18) sînt satisfăcute (avem $M_a = L_a = M_f = 1$, $L_f = o$ şi $L_f/(8-M_a)+L_aM_f/(8-M_a)^2 = 1/49 < 1$)

în timp ce condiţiile de convergenţă ale iteraţiei simple nu sînt satisfăcute (în acest caz $f(t,u) = u\sin(u)-1$ şi f nu este Lipschitz continuă în raport cu u).

Observaţia 3.8. Teorema (3.8) şi aplicaţia de la acest punct se găsesc în [58(2)]. Un rezultat similar relativ la ecuaţiile cu diferenţe (3.17) se poate obţine aplicînd teorema (3.9). Vom presupune că a şi f sînt mărginite şi că au derivate parţiale în raport cu u mărginite ; vom nota aceste margini cu M_a şi M_f şi respectiv M_a', M_f'. Vom presupune de asemenea că $M_a < 8$ şi că

$$\frac{M_f'}{8-M_a} + \frac{M_a' M_f}{(8-M_a)^2} < 1 \qquad (3.23)$$

Ca şi mai sus, $A(x)$ este o matrice inversabilă pentru orice $x \in \mathbb{R}^n$ şi $\|A(x)^{-1}\| \le (n+1)^2/(8-M_a)$ şi întrucît A şi Φ sînt evident diferenţiabile Fréchet pe $\mathbb{R}^n$, rămîne de verificat relaţia (3.20).

Deoarece $\|h^2 A_1^{-1} A_2(x)\| \le h^2(n+1)^2 M_a/8 = M_a/8 < 1$, rezultă că $A(x)^{-1} = \left[I + h^2 A_1^{-1} A_2(x)\right]^{-1} A_1^{-1} = \sum_{k=0}^{\infty} (-1)^k . h^{2k} . A_1^{-k} . (A_2(x))^k A_1^{-1}$.

Fie x^0 un element arbitrar din $\mathbb{R}^n$ şi fie $y^0 = A_1^{-1} \Phi x^0$. Avem :

$$\left[A(x)^{-1} . x^0\right]'_{x^0} = \sum_{k=1}^{\infty} (-1)^k . h^{2k} . A_1^{-k} \left[(A_2(x))^k y^0\right]'_{x^0}$$

$$\left\|\left[A(x)^{-1} \Phi x^0\right]'_{x^0}\right\| \le \sum_{k=1}^{\infty} h^{2k} \|A_1^{-1}\|^k \left\|\left[(A_2(x))^k y^0\right]'_{x^0}\right\|$$

Dar $\left[(A_2(x))^k y^0\right]'_{x^0} = \mathrm{Diag}\left[k a^{k-1}(x_1) a'(x_1) y_1^0, \ldots, k a^{k-1}(x_n) a'(x_n) y_n^0\right]$

astfel încît $\left\|\left[A(x)^{-1} \Phi x^0\right]'_{x^0}\right\| \le \|y^0\| M_a' \sum_{k=0}^{\infty} k . 8^{-k} M_a^{k-1} = \frac{8\|y^0\| M_a'}{(8-M_a)^2} \le \frac{M_a' M_f}{(8-M_a)^2}$.

Prin urmare, ţinînd cont că $\|\Phi'(x^0)\| \le h^2 M_f'$, obţinem :

$$\left\|\left[A(x)^{-1} \Phi x^0\right]'_{x^0} + A(x^0)^{-1} \Phi'(x^0)\right\| \le \left\|\left[A(x)^{-1} \Phi x^0\right]'_{x^0}\right\| +$$

$$+ \|A(x^0)^{-1}\| \|\Phi'(x^0)\| \le \frac{M_f'}{8-M_a} + \frac{M_a' M_f}{(8-M_a)^2}$$ astfel încît (3.20) este verificată ţinînd în vedere (3.23).

Observaţia 3.9. Dacă g şi f sînt mărginite şi cu derivate parţiale mărginite în raport cu u, rezultă că condiţiile (3.21) sînt satisfăcute cu $L_g = M_g'$ şi $L_f = M_f'$. Evident, dacă $M_g < 8$ şi (3.23) are loc, atunci este satisfăcută şi relaţia (3.22) astfel încît acest ultim rezultat este un corolar al rezultatului precedent.

3.4.3. Rezolvarea problemelor la limită slab neliniare cu ajutorul ecuaţiilor cu diferenţe finite corectate.

Vom demonstra următoarea teoremă similară cu teorema (3.8) dar pentru ecuaţii de forma (2.5).

Teorema 3.10. Dacă Φ din problema (2.5) este Lipschitz continuă cu constanta L_Φ şi există A^{-1} astfel încît $L_\Phi \| A^{-1} - qI \| < 1$, atunci metoda iteraţiilor simple

$$x^{k+1} = (A^{-1} - qI)\Phi x^k \qquad (3.24)$$

generează un şir convergent la x^* soluţia ecuaţiei (2.5).

Demonstraţie. Fie operatorul $Tx = (A^{-1} - qI)\Phi x$, atunci

$$\| Tx - Ty \| < L_\Phi \| A^{-1} - qI \| \| x - y \| < \| x - y \|$$

adică T este o contracţie. Conform principiului contracţiei ecuaţia (2.5) are o unică soluţie x^* şi procesul iterativ (3.24) converge la soluţia x^*.

Teorema 3.11. Fie $m > 0$ numărul punctelor diviziunii, dacă f din problema (2.2) este Lipschitz continuă cu constanta L_f astfel încît $L_f < 8 - 8/(3m^2 + 6m + 4)$ atunci procesul iterativ (3.24) converge la x^* soluţia problemei (2.5).

Demonstraţie. Considerăm norma $\| A \| = \max\limits_{i=\overline{1,m}} \sum\limits_{j=1}^{m} | a_{ij} |$, atunci

$$\| A^{-1} - qI \| = \| (I - qA)A^{-1} \| \leq \| I - qA \| \| A^{-1} \|$$

dar $\| I - qA \| < h^2/3+1$ iar $\| A^{-1} \| = (8h^2)^{-1}$. Dacă f este Lipschitz continuă cu constanta L_f, atunci şi Φ este Lipschitz

continuă cu constanta $L_{\Phi} = h^2 L_f$. Atunci $L_{\Phi}\|A^{-1} - qI\| <$ $< (h^2 + 3)L_f/24$. Relaţia $(h^2+3)L_f/24 < 1$ este echivalentă cu $L_f < 8-8/(3m^2 + 6m + 4)$. Deci dacă $L_f < 8-8/(3m^2+6m+4)$ atunci $L_{\Phi}\|A^{-1} - qI\| < 1$ deci conform teoremei (3.1o) procesul iterativ (3.24) converge la unica soluţie x^* a problemei (2.5).

Observaţia 3.1o. Dacă notăm cu $K_1(m) = 8-8/(3m^2+6m+4)$, atunci $K_1(1) = 96/13 \cong 7.38$ şi $K_1(9) = 2400/301 \cong 7.9734$. Evident că $K_1(m) \longrightarrow 8$ cînd $m \longrightarrow \infty$.

Observaţia 3.11. In mod analog putem obţine rezultate asemănătoare pentru ecuaţia (2.6) şi pentru procesul iterativ

$$x^{k+1} = (A^{-1} - qI + rA)\Phi x^k \qquad (3.25)$$

care converge la x^* soluţia ecuaţiei (2.6) dacă

$$L_f < 8 - 8(3(m+1)^2 + 1)/(15(m+1)^4 + 3(m+1)^2 + 1)$$

Dacă notăm cu $K_2(m) = 8 - 8(3(m+1)^2 + 1)/(15(m+1)^4 + 3(m+1)^2 + 1)$, atunci $K_2(1) = 640/87 \cong 7.35$, $K_2(9) \cong 400\,000/50187 \cong 7.9733$ şi evident $K_2(m) \longrightarrow 8$ cînd $m \longrightarrow \infty$.

Teorema 3.11. Fie $m^2 > 0$ numărul punctelor de diviziune din Δ (vezi paragraful 2.3.1). Dacă f din problema (2.15) este Lipschitz continuă pe D cu constanta

$$L_f < 24(m+1)^4 \sin^2(\pi/2(m+1))/(3(m+1)^2 - 2\sin^2(\pi/(2(m+1))))$$

atunci procesul iterativ

$$x^{k+1} = (A^{-1} - qI)\Phi x^k \qquad (3.26)$$

converge la x^* soluţia problemei (2.22).

Demonstraţie. Pentru $A \in \mathcal{M}_{m^2}(\mathbb{R})$ considerăm norma $\|A\| = \max_{i=1,m^2} |V_i|$ unde V_i sînt valorile proprii ale matricei A. Valorile proprii ale matricei A date de (2.17) sînt 41(2) , (15(4) :

$$V_{ij} = 4\sin^2(ih\pi/2) + 4\sin^2(jh\pi/2) \quad i,j = 1,m$$

atunci valorile proprii ale matricei A^{-1} sînt V_{ij}^{-1} deci :

$$A^{-1} = 1/8 \sin^2(h\pi/2)$$

Valorile proprii ale matricii $I-qA$ sînt $w_{ij} = 1-qv_{ij}$ $i,j = 1,m$, deci

$$\| I-qA \| = 1-2h^2\sin^2(h\pi/2)/3.$$

Dacă f este Lipschitz continuă cu constanta L_f atunci Φ este Lipschitz continuă cu constanta $L_\Phi = h^2L_f$. Ca procesul iterativ (3.26) să fie convergent conform teoremei (3.1o) va trebui ca $L_\Phi \| A^{-1} - qI \| < 1$ adică $h^2L_f \| I-qA \| \| A^{-1} \| < 1$ dar această relaţie este echivalentă cu relaţia :

$$L_f < 2(m+1)^4\sin^2(\pi/2(m+1))/(3(m+1)^2-2\sin^2(\pi/2(m+1))) \quad \forall m > o.$$

Observaţia 3.12. Dacă notăm cu $K_3(m) = 2(m+1)^4\sin^2(\pi/2(m+1))/(3(m+1)^2 - 2\sin^2(\pi/2(m+1)))$, avem că $K_3(m) \longrightarrow 2\pi^2 \cong 19.73$ cînd $m \to \infty$. Cîteva valori particulare ale lui K_3 : $K_3(1) = 192/11 \cong 17.45$, $K_3(2) \cong 972/53 \cong 18.34$.

Concluzie. Ecuaţiile cu diferenţe standard (2.4) sau (2.2o) au ordinul de aproximare $O(h^2)$, acest lucru înseamnă că soluţia ecuaţiei standard diferă de valoarea discretă corespunzătoare a soluţiei continue cu cel mult h^2. Acelaşi lucru se poate spune despre ecuaţiile cu diferenţe cu corecţii de ordinul 1 (2.3)

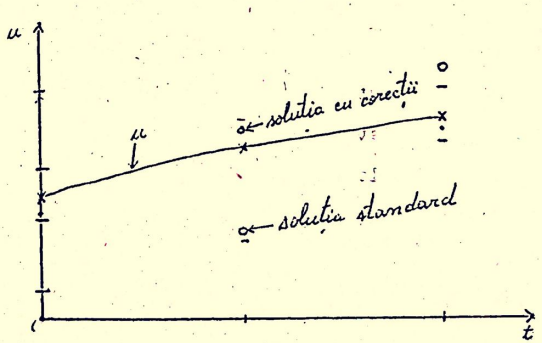


Fig.2.

şi (2.22) cu ordine de aproximare $O(h^4)$ şi despre ecuaţiile cu diferenţe cu corecţii de ordinul 2 (2.22) cu ordin de aproximare $O(h^6)$.

Dacă determinăm o soluţie cu metoda iteraţiilor simple, corespunzătoare uneia din ecuaţiile cu diferenţe, cu precizia ε înseamnă că două iteraţii succesive nu diferă între ele pe componente cu mai mult de valoarea ε. Deci impunerea unei precizii ε mari în determinarea soluţiilor ecuaţiei cu diferenţe nu ne duce la găsirea unor valori discrete care se apropie de valorile discrete ale soluţiei continue. Convergenţa spre valorile discrete ale soluţiei continue se obţine prin mărirea numărului de puncte ale diviziunii, adică $n \longrightarrow \infty$ sau $h \longrightarrow 0$, dar acest lucru are inconvenientul că măreşte dimensiunile ecuaţiilor cu diferenţe. In cazuri concrete este posibil ca numărul valorilor discrete ale soluţiei problemei continue determinantă cu ajutorul ecuaţiilor cu diferenţe standard să nu fie în concordanţă cu precizia soluţiilor ecuaţiilor cu diferenţe. De exemplu, să presupunem că avem nevoie de 9 valori echidistante ale soluţiei continue ale problemei (2.2) cu o precizie de 10^{-3}. Dacă apelăm la ecuaţiile cu diferenţe standard (2.4) şi determinăm soluţiile acestei probleme cu o precizie ε oricât de mică, tot nu putem obţine 9 valori discrete care să difere de cele 9 valori discrete ale soluţiei problemei continue cu mai puţin de 10^{-2}. Pentru a corecta această situaţie avem două rezolvări posibile : să facem pe $n = 99$ sau să apelăm la ecuaţiile cu diferenţe de ordinul 1 (2.5). Prima rezolvare, ($n = 99$), măreşte dimensiunile problemei de rezolvat considerabil, iar din soluţia obţinută avem nevoie numai de 10 valori. Cea de a doua rezolvare este mai avantajoasă pentru că volumul de calcul este rezonabil faţă de cazul standard, iar condiţiile de convergenţă nu diferă esenţial, după cum se ve-

de în teoremele (3.9),(3.10),(3.11), faţă de cazul standard. Din punct de vedere al programării pe calculator de asemenea a doua soluţie este mai convenabilă.

3.5. Metoda direcţiilor alternate

In acest paragraf se studiază o variantă a metodei direcţiilor alternate pentru ecuaţii neliniare [58(7)], variantă care păstrează anumite avantaje de calcul de la ecuaţiile cu diferenţe corespunzătoare problemelor la limită bilocale slab neliniare. Toate consideraţiile din acest paragraf au fost făcute pe spaţii finit dimensionale dar ele se pot transpune fără modificări în spaţii Hilbert reale.

3.5.1. Introducere. Metoda direcţiilor alternate a fost considerată în cazul liniar de D.Peaceman şi H.Rachford [54(1)] şi aplicată pentru rezolvarea numerică a ecuaţiilor cu diferenţe corespunzătoare problemelor la limită de tip eliptic sau parabolic. R.Kellogg [30(1)] a extins această metodă pentru cazul neliniar. Rezultatele sale le-a aplicat pentru rezolvarea ecuaţiei propagării radiaţiilor (ecuaţia căldurii) cu condiţii la limită neliniare de tipul Ştefan Boltzman [4(1)].

Metoda direcţiilor alternate comportă rezolvarea a două sisteme de ecuaţii neliniare la fiecare pas de iteraţie. In particular, în cazul ecuaţiilor cu diferenţe corespunzătoare problemelor la limită bilocale slab neliniare primul sistem este liniar cu matrice tridiagonală iar al doilea revine la rezolvarea unor ecuaţii cu câte o singură necunoscută fiecare. Evident, pentru probleme la limită neliniare de formă generală aceste avantaje nu se păstrează.

In acest paragraf se presupune o variantă a metodei direcţiilor alternate [58(7)] pentru ecuaţii neliniare de o formă parti-

culară, variantă în care primul sistem este întotdeauna liniar, păstrîndu-se astfel unul din avantajele metodei de la cazul slab neliniar.

Fie următoarea ecuaţie neliniară

$$A(x).x + \Phi x = 0 \qquad (3.27)$$

unde $x \in \mathbb{R}^n$, $x^T = (x_1, \ldots, x_n)$ iar $A : \mathbb{R}^n \to L(\mathbb{R}^n)$ şi $\Phi : \mathbb{R}^n \to \mathbb{R}^n$ sînt operatori neliniari. Metoda direcţiilor alternate pentru (3.17) dată în [3o(1)] va avea forma :

$$\begin{aligned} rx^{k+1/2} + A(x^{k+1/2})x^{k+1/2} &= rx^k - \Phi x^k \\ rx^{k+1} + \Phi x^{k+1} &= rx^{k+1/2} - A(x^{k+1/2})x^{k+1/2} \end{aligned} \qquad (3.28)$$

unde r este un număr real pozitiv. Ambele ecuaţii (3.28) sînt neliniare şi deci atît iteraţia intermediară $x^{k+1/2}$ cît şi iteraţia următoare x^{k+1} se obţin prin rezolvarea unor sisteme neliniare.

Varianta pe care o propunem rezultă din (3.28) înlocuind valoarea lui A în punctul $x^{k+1/2}$ cu valoarea lui A în punctul x^k, astfel încît (3.28) devine

$$\begin{aligned} rx^{k+1/2} + A(x^k)x^{k+1/2} &= rx^k - \Phi x^k \\ rx^{k+1} + \Phi x^{k+1} &= rx^{k+1/2} - A(x^k)x^{k+1/2} \end{aligned} \qquad (3.29)$$

Astfel prima ecuaţie este întotdeauna un sistem liniar cu matricea $rI + A(x^k)$.

3.5.2. Leme preliminare. Fie $\langle \cdot, \cdot \rangle$ şi $\| \cdot \|$ produsul scalar şi respectiv norma euclidiană pe spaţiul $\mathbb{R}^n$. Vom presupune că $A(x)$ este o matrice semipozitiv definită pentru orice $x \in \mathbb{R}^n$. Atunci $rI + A(x)$ este pozitiv definită şi prima ecuaţie (3.29) are soluţie unică în raport cu $x^{k+1/2}$. De asemenea, dacă Φ este un operator continuu şi monoton atunci $rI + \Phi$ este continuu şi strict monoton şi conform cu teorema de surjectivitate a lui

Minty (teorema 3.2), a doua ecuaţie (3.29) are de asemenea soluţie unică în raport cu x^{k+1}. Prin urmare, relaţiile (3.29) definesc în mod unic şirul $\{x^k\}$ pentru orice valoare iniţială $x^0 \in \mathbb{R}^n$.

Lema 3.7. Dacă $T : \mathbb{R}^n \to \mathbb{R}^n$ este continuu şi strict monoton, atunci T^{-1} există şi este Lipschitz continuu.

Demonstraţie. Existenţa lui T^{-1} este imediată. Fie $u,v \in \mathbb{R}^n$. Există $x,y \in \mathbb{R}^n$ astfel încît $Tx = u$, $Ty = v$, adică $T^{-1}u = x$, $T^{-1}v = y$. Din monotonia strictă a lui T rezultă

$$\| T^{-1}x - T^{-1}y \| \leq \frac{1}{c} \left\langle u-v , \frac{T^{-1}u-T^{-1}v}{T^{-1}u-T^{-1}v} \right\rangle \leq \frac{1}{c} \| u-v \|$$

astfel încît T^{-1} este Lipschitz continuu cu constanta Lipschitz $1/c$.

Fie B un operator continuu şi monoton şi fie r un număr real, pozitiv. Atunci există operatorul $(rI+B)^{-1}$ şi fie $T_B = (rI-B)(rI+B)^{-1}$. Următoarea lemă dată de Kellogg [30(1)] este esenţială pentru analiza convergenţei metodei direcţiilor alternate în cazul neliniar precum şi a variantei propuse.

Lema 3.8. Dacă B este continuu şi monoton atunci T_B este neexpansiv. Mai exact, dacă $x,y \in \mathbb{R}^n$ şi $x_1 = (rI+B)^{-1}x$, $y_1 = (rI+B)^{-1}y$ atunci

$$\| T_B x - T_B y \| \leq \frac{a-b}{a+b} \| x-y \|^2$$

unde $a = r^2 + \dfrac{\| Bx_1 - By_1 \|^2}{\| x_1 - y_1 \|^2}$, $b = 2r \dfrac{\langle Bx_1 - By_1, x_1 - y_1 \rangle}{\| x_1 - y_1 \|^2}$.

Să presupunem că $\langle A(x)u, u \rangle \geq \lambda \| u \|^2$ pentru orice $x,u \in \mathbb{R}^n$ şi fie $B = A(x)$; atunci pentru orice $x \in \mathbb{R}^n$ operatorul $T_{A(x)}$ este definit şi avem

$$a = r^2 + \frac{\| A(x)(x_1 - y_1) \|^2}{\| x_1 - y_1 \|^2} \leq r^2 + \| A(x) \|^2$$

$$b = 2r\,\frac{\langle A(x)(x_1-y_1),\ x_1-y_1\rangle}{\|x_1-y_1\|^2} \geqslant 2r\lambda$$

se poate vedea uşor că

$$\frac{a-b}{a+b} \leq \frac{r^2+\|A(x)\|^2-2r\lambda}{r^2+\|A(x)\|^2+2r\lambda}$$

Dacă presupunem acum că $\|A(x)\| \leq \beta$, pentru orice $x\in\mathbb{R}^n$, atunci

$$\frac{a-b}{a+b} \leq \frac{r^2+\beta^2-2r\lambda}{r^2+2r\lambda} = \alpha \qquad (3.30)$$

Pentru r suficient de mare $\alpha < 1$ şi $T_{A(x)}$ este o contracţie în $\mathbb{R}^n$.

Lema 3.9. Să presupunem că $A(x)$ este o matrice pozitiv definită şi că $\|A(x)\| \leq \beta$ pentru orice $x\in\mathbb{R}^n$ şi fie $r > \beta$. Atunci

$$\|T_{A(x)} - T_{A(y)}\| \leq \frac{2r}{(r-\beta)^2}\,\|A(x)-A(y)\| .$$

Demonstraţie.

$$\|T_{A(x)}-T_{A(y)}\| \leq \|(rI-A(x))(rI+A(x))^{-1}-(rI-A(x))(rI+A(y))^{-1}\| +$$
$$\|(rI-A(x))(rI+A(y))^{-1}-(rI-A(y))(rI+A(y))^{-1}\| \leq$$
$$\|rI-A(x)\|\,\|(rI+A(x))^{-1}-(rI+Ay)^{-1}\| +$$
$$+\|(rI+A(y))^{-1}\|\,\|A(y)-A(x)\|$$

Ţinînd cont că $\|r^{-1}A(y)\| \leq r^{-1}\beta < 1$ rezultă că:

$$(rI+A(y))^{-1} = r^{-1}.(I+r^{-1}A(y))^{-1} = r^{-1}\sum_{k=0}^{\infty}(-1)^k r^{-k}A(y)^k \text{ şi deci}$$

$$\|(rI+A(y))^{-1}\| \leq r^{-1}\sum_{k=0}^{\infty}(r^{-1}\beta)^k = \frac{1}{r-\beta}$$

De asemenea, printr-un calcul simplu obţinem :

$$\|(rI+A(x))^{-1}-(rI+A(y))^{-1}\| \leq r^{-2}\,\|A(x)-A(y)\|\sum_{k=0}^{\infty}k(r^{-1}\beta)^{k-1} =$$
$$= \frac{1}{(r-\beta)^2}\,\|A(x)-A(y)\|$$

Prin urmare

$$\|T_{A(x)}-T_{A(y)}\| \le \frac{1}{(r-\beta)^2}\|A(x)-A(y)\|\,\|rI-A(x)\| +$$

$$+\frac{1}{r-\beta}\|A(x)-A(y)\| \le \frac{2r}{(r-\beta)^2}\|A(x)-A(y)\|$$

3.5.3. Analiza convergenţei

Studiul convergenţei metodei (3.29) are la bază ideile din [3o(1)], fără însă ca demonstraţiile din lucrarea amintită să se poată transpune în mod banal la cazul nostru. Fie $y^k=(rI+\Phi)x^k$, $y^{k+1/2}=(rI+A(x^k))x^{k+1/2}$. Din (3.29) rezultă :

$$y^{k+1/2} = T_{\Phi}\cdot y^k \quad , \quad y^{k+1} = T_{A(x^k)}\, y^{k+1/2}$$

Vom presupune că ecuaţia (3.27) are o soluţie $x^*\in\mathbb{R}^n$ şi fie $x^* = (rI+\Phi)x^*$, $x^*_A = (rI+A(x^*))x^*$. Deoarece $A(x^*).x^* = -\Phi x^*$ rezultă că $x^*_\Phi = T_{A(x^*)}x^*_A$, $x^*_A = T_\Phi\, x^*_\Phi$.

Teorema 3.12. Presupunem că ecuaţia (3.27) are o soluţie $x^*\in\mathbb{R}^n$ şi că, A şi Φ satisfac următoarele condiţii

1) matricea $A(x)$ este pozitiv definită $\langle A(x)u,u\rangle \ge \lambda\|u\|^2$ pentru orice $x,u\in\mathbb{R}^n$

2) $\|A(x)\|\le\beta$ pentru orice $x\in\mathbb{R}^n$

3) $\|A(x)-A(y)\|\le L\|x-y\|$ pentru orice $x,y\in\mathbb{R}^n$

4) Φ este continuu şi monoton

5) constantele λ şi L verifică inegalitatea: $2\lambda < L\|x^*\|$

Atunci pentru r suficient de mare, iteraţia (3.29) defineşte un şir unic $\{x^k\}$ care converge la x^*.

Demonstraţie. Existenţa şi unicitatea şirului $\{x^k\}$ au fost realizate la punctul 3.5.4. Rămîne să arătăm convergenţa şirului $\{x^k\}$ la x^*. Din lemele (3.8) şi (3.9) şi din condiţia 3) rezultă :

$$\|y^{k+1}-x^*\| = \|T_{A(x^k)}y^{k+1/2} - T_{A(x^*)}x^*_A\| \le \|T_{A(x^k)}y^{k+1/2} - T_{A(x^k)}x^*_A\| +$$

$$+\|T_{A(x^k)}\cdot x_A^* - T_{A(x^*)}x_A^*\| \le \alpha\|y^{k+1/2}-x_A^*\| + L\|x_A^*\|\frac{2r}{(r-\beta)^2}\|x^k-x^*\| =$$
$$=\alpha\|T_{\Phi}y^k - T_{\Phi}x_{\Phi}^*\| + L\|x_A^*\|\frac{2r}{(r-\beta)^2}\,(rI+\Phi)^{-1}y^k-(rI+\Phi)^{-1}x^* .$$

Din condiția 4) rezultă că $rI+\Phi$ este continuu și strict monoton cu constanta r iar din lema (3.7) rezultă că $(rI+\Phi)^{-1}$ este Lipschitz continuu cu constanta $1/r$. Ținînd cont de acest fapt și de lema (3.8) avem :

$$\|y^{k+1}-x_{\Phi}^*\| \le \left[\alpha + L\|x_A^*\|\frac{2}{(r-\beta)^2}\right]\|y^k-x_{\Phi}^*\| \le \left[\alpha+2L\|x^*\|\frac{r+\beta}{(r-\beta)^2}\right]\|y^k-x^*\|$$

Să notăm cu α_1 constanta $\alpha + 2L\|x^*\|(r+\beta)/(r-\beta)^2$, unde α este dat de (3.3o). Vom impune condiția $\alpha_1 < 1$ și atunci șirul $\{y^k\}$ converge la x_{Φ}^* ; dar această condiție este echivalentă cu

$$(4\lambda - 2L\|x^*\|)r^3 + a_1r^2 + a_2r + a_3 > 0$$

unde a_1 , a_2 , a_3 depind de λ , β , L, $\|x^*\|$. Pentru r suficient de mare această condiție este satisfăcută întrucît $2\lambda > L\|x^*\|$ conform ipotezei 5). Prin urmare $\alpha_1 < 1$ și deci $y^k \longrightarrow x_{\Phi}^*$ cînd $k \longrightarrow \infty$. Acum deoarece $x^k = (rI+\Phi)^{-1}y^k$ și $(rI+\Phi)^{-1}$ este operator continuu rezultă că $x^k \longrightarrow (rI+\Phi)^{-1}x_{\Phi}^* = x^*$.

3.6. Metode de tip gradient

In acest paragraf se studiază convergența unor metode de tip gradient utilizînd în mod sistematic conceptul de operator cvasi-neexpansiv (definiția 3.2). Pe această bază se demonstrează convergența locală a unei metode de tip gradient generalizată [59(2)] și se obțin în particular unele rezultate cunoscute [1(1)], [21(1)] [25(1)]. Menționăm că rezultatul principal din acest paragraf (teorema 3.13) constituie generalizarea teoremei 2 din [41(5)].

3.6.1. Introducere. In lucrarea [41(4)] utilizînd conceptul de operator cvasi-neexpansiv, s-au obținut condiții de convergență locală a metodei lui Newton și a metodei gradientului pentru ecuații neliniare pe spații finit dimensionale. Condițiile obținute

în acest mod pentru metoda lui Newton sînt asemănătoare cu cele din teoremele bine cunoscute de convergenţă globală ale lui Kantorovici [28(1)(3)]; pentru metoda gradientului condiţiile obţinute sînt de acelaşi tip.

Vom considera în acest paragraf ecuaţii de forma $Fx = o$, unde $F: D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$ este un operator neliniar (pe spaţiile $\mathbb{R}^n, \mathbb{R}^m$ vom considera normele euclidiene iar D este o mulţime deschisă din $\mathbb{R}^m$). După cum se ştie [48(1)] metoda gradientului are forma :

$$x^{k+1} = x^k - \lambda_k F'(x^k)^T Fx^k \quad , \quad k = o,1,...$$

unde λ_k reprezintă "lungimea pasului" (steep lenght). Din punct de vedere geometric această metodă revine la deplasări succesive pe direcţia gradientului funcţionalei $f: \mathbb{R}^n \longrightarrow \mathbb{R}$, $f(x) = \|Fx\|^2$ cu pasul λ_k.

In cele ce urmează vom considera iteraţia mai generală

$$x^{k+1} = x^k - \|F'(x^k)\|^2 F'(x^k)^T A(x^k) Fx^k, \quad k = o,1,... \quad (3.31)$$

unde $A: \mathbb{R}^n \longrightarrow L(\mathbb{R}^m)$ este un operator neliniar, iteraţie pe care o vom numi metoda gradientului generalizată. După cum se ştie metoda gradientului are convergenţă liniară ; alegînd în mod convenabil operatorul A se pot obţine metode cu ordin de convergenţă superior. De exemplu, dacă se ia $A(x) = \left[F'(x)F'(x)^T\right]^{-1} \|F'(x)\|^2$ atunci (3.31) coincide cu metoda lui Newton care are, după cum se ştie, convergenţă pătratică.

3.6.2. Analiza convergenţei. Vom demonstra mai întîi o lemă simplă de tipul formulelor de medie din analiza n-dimensională.

Lema 3.1o. Fie D o mulţime deschisă din $\mathbb{R}^n$ şi $F: D \longrightarrow \mathbb{R}^m$ un operator diferenţiabil Fréchet pe D. Atunci

$$Fy - Fx = \left[F'(x) + \delta(x,y)\right](y-x) \text{ pentru } \forall\, x,y \in D \quad (3.32)$$

cu $\delta(x,y) \longrightarrow o$, $y \longrightarrow x$.

Demonstrație

Plecăm de la identitatea evidentă

$$Fy - Fx = \left[F'(x)+(Fy-Fx-F'(x)(y-x))\frac{(y-x)^T}{\|y-x\|^2}\right](y-x)$$

și luăm $\delta(x,y)= Fy-Fx-F'(x)(y-x)\frac{(y-x)^T}{\|y-x\|^2}$, se vede că $\|\delta(x,y)\|=$
$= \|Fy-Fx-F'(x)(y-x)\| / \|y-x\| \longrightarrow 0$, $y \longrightarrow x$.

Teorema 3.13. Fie $F: D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$ un operator neliniar și $x^* \in$
$\in D$ o soluție a ecuației $Fx = o$. Presupunem că F este diferențiabil Fréchet pe $S = S(x^*,r) \subset D$, aplicația $G : D \longrightarrow$
$\longrightarrow L(\mathbb{R}^m,\mathbb{R}^n)$ este de rangul n, pentru orice $x \in S$. Fie $A :$
$D \setminus \{x^*\} \longrightarrow L(\mathbb{R}^m)$ un operator continuu care satisface relația

$$\|A(x) - I\| \leq \eta < 1 \quad \text{pentru orice} \quad x \in S \setminus \{x^*\} . \qquad (3.33)$$

Atunci x^* este un punct de atracție pentru iterația (3.31).

Demonstrație. Deoarece $F'(x)$ este de rang n rezultă că
$\|F'(x)\| \neq o$ pe S. Fie atunci operatorul $T: S\setminus\{x^*\} \longrightarrow \mathbb{R}^n$
definit de

$$Tx = x - \|F'(x)\|^{-2}F'(x)^T A(x)Fx$$

Ținînd cont de (3.32) avem :

$$Tx-x^* = x-x^*-\|F'(x)\|^{-2}F'(x)^T A(x)\left[F'(x)+\delta(x,x^*)\right](x-x^*) =$$
$$=\left[I-\|F'(x)\|^{-2}F'(x)^T A(x)F'(x)-\|F'(x)\|^{-2}F'(x)^T A(x)\delta(x,x^*)\right](x-x^*)$$

Pentru prescurtarea scrierii vom nota $F'(x)= F'$, $A(x)= A$,
$\delta(x,x^*) = \delta$. Obținem

$$\|Tx-x^*\| \leq \left(\|I-\|F'\|^{-2}(F')^T AF'\| + \|F'\|^{-2}\|A\|\,\|\delta\|\right)\|x-x^*\| \qquad (3.34)$$

Fie $z \in \mathbb{R}^n$, $\|z\| = 1$. Avem :

$$\|z-\|F'\|^{-2}(F')^T AF'z\|^2 = 1-2\|F'\|^{-2}\langle F'z, AF'z\rangle + \|F'\|^{-2}\|AF'z\|^2 =$$
$$= 1-\|F'\|^{-2}(\|F'z\|^2-\|(A-I)F'z\|^2) = 1-(1-\|A-I\|^2)\|F'\|^{-2}\|F'z\|^2$$

Ținînd cont de (3.33) rezultă :

$$\|z-\|F'\|^{-2}(F')^T AF'z\|^2 \leq 1-(1-\eta^2)\|F'\|^{-2}\|F'z\|^2.$$

Să observăm că $\|F'(x)\| \leq \beta_1$ pentru orice $x \in S_1(x^*,r) \subset S$ și deci $\|F'(x)\|^{-2} \geq \beta_1^{-2}$ apoi că $\|F'(x)z\| \geq \beta_2 > 0$ pentru orice $x \in S$ și $\|z\| = 1$ (în caz contrar $F'(x)$ nu ar avea rangul n). Prin urmare :

$$\|I-\|F'\|^{-2}(F')^T A F'\| = \max_{\|z\|=1} \|z-\|F'\|^{-2}(F')^T A F' z\| \leq \left[1-(1-\eta^2)\beta_1^{-2}\beta_2^2\right]^{1/2} \quad (3.35)$$

Să mai observăm că $\|F'(x)\| \geq \beta_2$ și deci $\|F'(x)\|^{-1} \leq \beta_2^{-1}$ pentru orice $x \in S$. Din (3.33) rezultă că $\|A(x)\| \leq 1+\eta$ pentru orice $x \in S \setminus \{x^*\}$. Să notăm cu $\beta = \left[1-(1-\eta^2)\beta_1^{-2}\beta_2^2\right]^{1/2}$. Conform lemei (3.10) $\delta(x,x^*) \longrightarrow 0$, $x \longrightarrow x^*$ și deci putem determina o constantă r_2 astfel încît $S(x^*,r_2) \subset S_1$ și

$$\|\delta\| \leq \frac{(1-\beta)\beta_2}{1+\eta} \quad \text{pentru orice} \quad x \in S(x^*,r_2) - \{x^*\}$$

așadar $\|F'\|^{-1}\|A\|\|\delta\| < 1-\beta$ pentru orice $x \in S(x^*,r_2)$. Ținînd cont acum și de (3.35) avem :

$$\|I-\|F'\|^{-2}(F')^T A F'\| + \|F'\|^{-1}\|A\|\|\delta\| < 1.$$

Astfel încît din relația (3.34) rezultă

$$\|Tx-x^*\| < \|x-x^*\| \quad \text{pentru orice } x \ S(x^*,r_2)$$

adică T este cvasi-neexpansiv pe $S(x^*,r_2) \setminus \{x^*\}$ și se aplică lema 3.3.

Observația 3.13. Dacă operatorul A este definit și continuu în x^*, condiția (3.33) se poate înlocui cu

$$\|A(x^*) - I\| < 1.$$

In continuare vom aplica teorema (3.13) pentru a obține condiții de convergență locală pentru unele metode iterative cunoscute.

3.6.3. Metoda gradientului

Dacă în (3.31) luăm $A = I$ obținem metoda gradientului cu $\lambda_k = \|F'(x^k)\|^{-2}$, adică :

$$x^{k+1} = x^k - \|F'(x^k)\|^{-2} F'(x^k)^T F x^k \qquad (3.36)$$

Condiţia (3.33) este evident satisfăcută şi din teoreme (3.13) rezultă

Corolarul 1. Fie $F : D \subset \mathbb{R}^n \to \mathbb{R}^m$ un operator neliniar şi $x^* \in D$ o soluţie a ecuaţiei $Fx = o$. Presupunem că F este diferenţiabil Fréchet pe $S(x^*,r) \subset D$ şi că matricea $F'(x)$ este mărginită pe $S(x^*,r)$ şi de rang n. Atunci x^* este punct de atracţie pentru iteraţia (3.36).

Metoda gradientului a fost considerată printre primii de H.Curry [16(1)] precum şi de M.Vainberg [72(3),(4)], M.Altman [1(1),(2)], M.Nashed [45(2)]. Sub forma (3.36 şi pentru cazul particular $n = 1$, această metodă a fost considerată de M.Altman [1(1),(2)]. De asemenea metoda gradientului a fost considerată în [50(1)], [41(1),(2)] pentru cazul $n = m$, obţinîndu-se condiţii de convergenţă substanţial mai tari. Astfel se cere ca F să fie diferenţiabil Fréchet pe o sferă $S(x^*,r)$ şi să satisfacă următoarele condiţii :

1) matricea $F'(x)$ să fie inversabilă şi $\| F'(x)\|^{-1} \leq \beta$, pentru orice $x \in S(x^*,r)$

2) $\| F'(x) - F'(y)\| \leq \gamma \| x - y\|$ pentru orice $x,y \in S(x^*,r)$

3) $\beta \gamma r < 2$.

Sub o formă asemănătoare, dar cu condiţii de asemenea mai tari, corolarul 1 apare în lucrarea [41(1)].

3.6.4. Metoda Newton - gradient combinată.

Fie din nou $F : D \subset \mathbb{R}^n \to \mathbb{R}^m$ un operator neliniar, f_i, $i = \overline{1,m}$ componentele lui F şi f_i' derivatele Fréchet ale componentelor, deci $f_i' = (\partial f_i / \partial x_1, \ldots, \partial f_i / \partial x_n)$ unde x_j, $j = \overline{1,n}$ sînt componentele unui element $x \in \mathbb{R}^n$. Vom presupune

că $\|f_i'(x)\| \neq 0$, $i = \overline{1,m}$, $x \in D$. Luăm acum pe A din teorema (3.15) de forma

$$A(x) = -\lambda \|F'(x)\|^2 B(x)^{-1} \text{ , unde}$$

$$B(x) = \begin{pmatrix} \|f_1'(x)\|^2 & & 0 \\ & \ddots & \\ 0 & & \|f_m'(x)\|^2 \end{pmatrix}$$

iar λ este un număr pozitiv. Obținem următoarea metodă iterativă, numită metoda Newton - gradient combinată [25(1)]

$$x^{k+1} = x^k - \lambda F'(x^k)^T B(x^k)^{-1} F(x^k).$$

Ținînd cont că

$$A(x) - I = \begin{pmatrix} \lambda \dfrac{\|F'(x)\|^2}{\|f_1'(x)\|^2} - 1 & & 0 \\ & \ddots & \\ 0 & & \lambda \dfrac{\|F'(x)\|^2}{\|f_m'(x)\|^2} - 1 \end{pmatrix},$$

rezultă $\| A(x) - I \| = \max_i \left| \lambda \dfrac{\|F'(x)\|^2}{\|f_i'(x)\|^2} - 1 \right|$

Astfel încît condiția (3.33) este satisfăcută pe $S = S(x^*,r) \subset D$, dacă

$$\lambda < \frac{2\|f_i'(x)\|^2}{\|F'(x)\|^2} \quad \text{pentru } i = \overline{1,m} \text{ pentru orice } x \in S \quad (3.37)$$

Din teorema (3.15) obținem :

Corolarul 2. Fie $F: D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^m$ un operator neliniar, f_i , $i = \overline{1,m}$ componentele lui F și $x^* \in D$ o soluție a ecuației $Fx = 0$. Presupunem că F este diferențiabil Fréchet pe $S = S(x^*,r) \subset D$ și că matricea $F'(x)$ este mărginită pe S și de rang n ; presupunem de asemenea că $\|f_i'(x)\| \neq 0$ pe S, $i =$

= 1,m şi că satisface (3.37). Atunci x^* este punct de atracţie pentru metoda Newton - gradient combinată.

Această metodă a fost considerată de H.Hart şi T.Motzkin [25(1)]. Corolarul 2 este identic cu teorema centrală din lucrarea amintită (teorema 5.1) pentru cazul neliniar cu excepţia limitei superioare pentru λ care în [25(1)] este dată de $1/(\lambda_1 + \ldots + \lambda_m)$, λ_i fiind valorile proprii ale matricei $F'(x^*)^T F'(x^*)$.

3.6.5. Metoda lui Fridman.

Să luăm acum pe A de forma :

$$A(x) = \frac{\|F'(x)\|^2 \|F(x)\|^2}{\|F'(x)^T Fx\|^2} \cdot I \qquad (3.38)$$

Se vede că, în acest caz, dacă F este diferenţiabil Fréchet pe $S = S(x^*, r) \subset D$, atunci A este un operator diagonal definit pe $S \setminus \{x^*\}$. Metoda gradientului generalizată (3.31) va avea forma :

$$x^{k+1} = x^k - \frac{\|Fx^k\|^2}{\|F'(x^k)^T Fx^k\|^2} F'(x^k)^T Fx^k \qquad (3.39)$$

Să verificăm condiţia (3.33). Avem :

$$\|A(x) - I\| = \left| \frac{\|F'(x)\|^2 \|Fx\|^2}{\|F'(x)^T Fx\|^2} - 1 \right.$$

astfel încît (3.33) este satisfăcută dacă

$$\frac{\|F'(x)\|^2 \|Fx\|^2}{\|F'(x)^T Fx\|^2} \leq \eta^2 < 2 \quad \text{pentru orice } x \in S.$$

Să presupunem că $F'(x)$ este mărginită pe S, $\|F'(x)\| \leq \beta_1$. Vom impune condiţia

$$\frac{\|F'(x)^T z\|}{\|z\|} \geq m > 0 \quad \text{pentru orice } x \in S,\ z \in \mathbb{R}^m \qquad (3.40)$$

şi vom cere ca m să satisfacă relaţia $\beta_1 m^{-1} < \sqrt{2}$. In acest caz A dat de (3.38) satisface relaţia (3.35). Să observăm de asemenea că în condiţia (3.40) matricea $F'(x)$ are rangul n.

Pe baza teoremei (3.13) obţinem următorul rezultat asupra metodei (3.39) :

Corolarul 3. Fie $F: D \subset \mathbb{R}^n \to \mathbb{R}^m$ un operator neliniar şi $x^* \in D$ o soluţie a ecuaţiei $Fx = 0$. Presupunem că F este diferenţiabil Fréchet pe $S = S(x^*, r) \subset D$ şi că matricea $F'(x)$ este mărginită pe S ($\|F'(x)\| \leq \beta_1$, pentru orice $x \in S$) şi satisface relaţia (3.40). Presupunem de asemenea că constantele β_1 şi m verifică relaţia $\beta_1 m^{-1} < \sqrt{2}$. Atunci x^* este punct de atracţie pentru iteraţia (3.39).

Metoda (3.39) a fost considerată de V.Fridman [21(1),(2)] pentru rezolvarea ecuaţiilor (liniare sau neliniare) în spaţii Hilbert reale. Pentru cazul neliniar condiţiile de convergenţă sînt asemănătoare cu cele din corolarul 3. Să observăm că (3.39) coincide cu metoda gradientului considerată la punctul 3.6.1. în care pasul λ_k are valoarea $\|Fx^k\|^2 / \|F'(x^k)^T Fx^k\|^2$. In lucrarea amintită V.Fridman a dedus această valoare din condiţia de minimizare a funcţionalei $\|x^k - F'(x^k)Fx^k\|^2$ după parametrul λ_k, făcîndu-se apoi aproximarea $F'(x^k)(x^k - x^*) \simeq Fx^k$.

3.6.6. Observaţii asupra metodei gradientului generalizată cu ordin superior de convergenţă. După cum am văzut (punctul 3.6.1) operatorul A se poate alege astfel încît metoda gradientului generalizată să coincidă cu metoda lui Newton. Să luăm deci $n = m$ şi

$$A(x) = \|F'(x)\|^2 (F'(x)^T)^{-1} . F'(x)^{-1}$$

iteraţia (3.31) se reduce în acest caz la metoda cunoscută a lui Newton.

Vom presupune că

$$\|F'(x)\| \, \|F'(x)^{-1}\| \leq \eta < \sqrt{2} \quad \text{pentru orice } x \in S(x^*, r)$$

unde r este un număr real pozitiv şi fie $z \in \mathbb{R}^n$, $\|z\| = 1$. Avem

$$\|z - \|F'(x)\|^2 (F'(x)^T)^{-1} F'(x)^{-1} z\|^2 \leq 1 - 2\|F'(x)\|^2 < z, (F'(x)^{-1})^T F(x)^{-1} z >$$

$$+\|F'(x)\|^4.\|F'(x)^{-1}\|^2.\|F(x)^{-1}z\|^2 =$$
$$= 1 - \|F'(x)\|^2\|F'(x)^{-1}z\|^2(2 - F'(x)\|^2\|F'(x)^{-1}\|).$$

Dacă mai presupunem că $F'(x)$ este o matrice inversabilă pentru orice $x \in S(x^*,r)$, atunci $\|F'(x)\|, \|F'(x)^{-1}z\|$ sînt mărginite inferior (fie β_1 și β_2 constantele pozitive respective) și prin urmare

$$\|I - \|F'(x)\|^2(F'(x)^T)^{-1}F(x)^{-1}\|^2 \leq 1 - \beta_1^2\beta_2(2-\gamma^2) < 1$$

relație care constituie condiția (3.33) din teorema (3.13). Rezultă de aici că x^* este punct de atracție pentru metoda lui Newton. Dar acest rezultat este în mod practic neinteresant, întrucît se știe că metoda lui Newton este local convergentă cu singurele condiții ca F să fie diferențiabil Fréchet în punctul x^* iar matricea $F'(x^*)$ să fie nesingulară.

3.7. O metodă iterativă specială pentru ecuații de forma (2.12).

In [41(6),(1),(2),(5)] s-a studiat o metodă iterativă specială pentru rezolvarea ecuațiilor neliniare de forma (2.12) metodă care prezintă anumite avantaje de programare (paragraful 4.4). In acest paragraf se generalizează [58(4)] unele rezultate din [41(6),(1),(2),(5)] prin slăbirea condițiilor de convergență. In particular, se obține metoda liniilor paralele, de asemenea cu condiții de convergență mai slabe.

3.7.1. Algoritmul de calcul și analiza convergenței. Vom considera ecuații de forma (2.12) deci de forma :

$$A(x).x + \Phi x = 0 \qquad (3.41)$$

unde $A: D \subset \mathbb{R}^n \longrightarrow L(\mathbb{R}^n)$ și $\Phi : D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ sînt operatori neliniari. Vom presupune că ecuația (3.41) are o soluție $x^* \in D$ și vom considera următorul proces iterativ pentru aproximarea soluției,

$$x^{k+1} = x^k - \lambda A(x^k)^T[A(x^k)x^k + \Phi x^k] \qquad (3.42)$$

fiind o constantă reală pozitivă.

Observaţia 3.14. In esenţă această metodă este o variantă a metodei gradientului pentru ecuaţii de forma (3.41). Dacă A şi Φ sînt operatori constanţi (3.42) coincide cu metoda gradientului pentru ecuaţii liniare, considerată de numeroşi autori [57(1)], [45(2)], [18(1)].

Observaţia 3.15. Metoda iterativă (3.42) a fost considerată în lucrările [41(6),(1),(2),(5)] unde s-au dat condiţii de convergenţă locală iar în cazuri particulare s-au obţinut unele rezultate cunoscute.

Vom considera pe spaţiul $\mathbb{R}^n$ norma euclidiană

Teorema 3.14. Presupunem că operatorii A şi Φ sînt diferenţiabili Fréchet în punctul x^*, soluţie a ecuaţiei (3.41) şi că matricea $A(x^*)$ este nesingulară. Presupunem că este îndeplinită următoarea condiţie

$$\|A(x)^{-1}\| \, \| \left[A(x)x^*\right]'_{x^*} + \Phi'(x^*)\| < 1 \quad \text{şi că} \quad \lambda \leq \|A(x^*)\|^{-2} \tag{3.43}$$

Atunci x^* este punct de atracţie pentru iteraţia (3.42).

Demonstraţie. Fie $T: \mathbb{R}^n \longrightarrow \mathbb{R}^n$ definit de

$$Tx = x - \lambda . A(x)^T \left[A(x)x + \Phi x\right]$$

utilizînd lema (3.2) avem :

$$T'(x^*) = I - \lambda A(x^*)^T \left\{ A(x^*) + \left[A(x)x^*\right]'_{x^*} + \Phi'(x^*) \right\} \tag{3.44}$$

Deoarece $\lambda \|A(x^*)\|^2 \leq 1$ obţinem pentru orice $y \in \mathbb{R}^n$, $\|y\| = 1$

$$\|T'(x^*)y\|^2 = \|y - A(x^*)^T \left\{ A(x^*) + \left[A(x)x^*\right]'_{x^*} + \Phi'(x^*) \right\} y\|^2 \leq$$

$$\leq 1 - 2\lambda < \left\{ A(x^*) + \left[A(x)x^*\right]'_{x^*} + \Phi'(x^*) \right\} y, A(x^*)y > + \lambda \|A(x^*)y\| +$$

$$+ \left\{ \left[A(x)x^*\right]'_{x^*} + \Phi'(x^*) \right\} y\|^2 \leq 1 - \lambda \|A(x^*)y\|^2 + \lambda \| \left[A(x)x^*\right]'_{x^*} +$$

$$+ \Phi'(x^*)\|^2 \leq 1 - \lambda (\|A(x^*)^{-1}\|^{-2} - \| \left[A(x)x^*\right]'_{x^*} + \Phi'(x^*)\|^2 .$$

Ţinînd cont de condiţia (3.43) rezultă că $\|T'(x^*)\| < 1$ şi con-

vergență locală a metodei rezultă din teorema (3.1).

Observația 3.16. In general $\| T'(x^*) \| \neq 0$ și metoda (3.42) are numai convergență liniară. Acest fapt poate provoca în cazuri particulare creșterea substanțială a volumului de timp calculator, chiar dacă se lucrează exclusiv în memoria internă (paragraful 4.4).

Putem da și următoarea variantă a teoremei 3.14 :

Teorema 3.15. Fie A și Φ ca în teorema 3.14. In locul relației (3.43) vom considera că este satisfăcută relația :

$$\langle A(x^*)y, \{[A(x)x^*]'_{x^*} + \Phi'(x^*)\}y \rangle \gg 0 \quad \text{pentru orice } y \in \mathbb{R}^n \qquad (3.45)$$

De asemenea, presupunem că λ satisface relațiile

$$\lambda \leq \|A(x^*)\|^{-2}$$
$$\lambda < \|A(x^*)^{-1}\|^{-2} \cdot \|A(x^*)\|^{-2} \|[A(x)x^*]'_{x^*} + \Phi'(x^*)\|^{-2} \qquad (3.46)$$

Atunci x^* este punct de atracție pentru iterația (3.42).

Demonstrație. Utilizînd expresia (3.44) a derivatei $T(x^*)$ putem scrie :

$$\| T'(x^*)y\|^2 \leq 1 - \lambda \|A(x^*)y\|^2 - 2\lambda \langle A(x^*)y, \{[A(x)x^*]'_{x^*} + \Phi'(x^*)\}y \rangle +$$
$$+ 2\lambda^2 \|A(x^*)\|^2 \langle A(x^*)y, \{[A(x)x^*]'_{x^*} + \Phi'(x^*)\}y \rangle +$$
$$+ \lambda^2 \|A(x^*)\|^2 \|[A(x)x^*]'_{x^*} + \Phi'(x^*)\|^2.$$

Acum ținînd cont de (3.45) și (3.46) rezultă

$$\| T'(x^*)y\|^2 \leq 1 - \lambda(\|A(x^*)^{-1}\|^{-2} - \lambda \|A(x^*)\|^2 \|[A(x)x^*]'_{x^*} + \Phi'(x^*)\|^2) < 1$$

și se aplică din nou teorema (3.1).

Observația 3.17. Teoremele 3.15 și 3.16 sînt prezentate în [41(1),(2)] pentru cazul unui spațiu Hilbert real de dimensiune finită și cu condiții de convergență mai tari. Astfel se cere ca operatorii A și Φ să fie diferențiabili Fréchet pe o anumită sferă centrată în x^* și ca derivatele acestor operatori să fie mărginite.

3.7.2. Metoda liniilor paralele

Pe baza teoremei (3.15) se poate obține o teoremă de convergență locală pentru metoda liniilor paralele.

Fie $D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$ un operator neliniar și x^* o soluție a ecuației $Fx = o$. Pentru a putea aplica teorema (3.15) vom scrie această ecuație sub forma $\lambda x + (Fx - x) = o$ astfel încît $A(x) = I$ iar $\phi x = Fx - x$. Deoarece $\phi'(x) = F'(x) - I$ condiția (3.45) devine :

$$\langle F'(x^*)y, y \rangle \geq \| y \|^2 \quad \text{pentru orice } y \in \mathbb{R}^n \qquad (3.47)$$

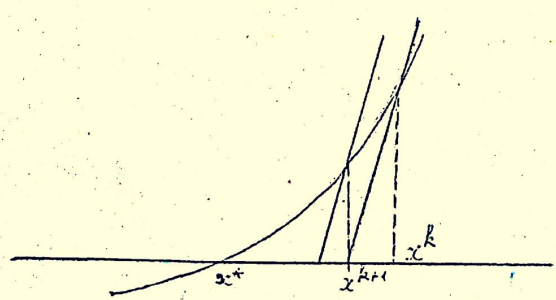
iar condițiile impuse lui λ (3.46) se reduc la :

$$\lambda \leq 1 \ , \ \lambda < \| F'(x^*) - I \|^{-2} \qquad (3.48)$$

Iterația (3.42) va avea forma $x^{k+1} = x^k - \lambda Fx^k$, iterație cunoscută sub numele de metoda liniilor paralele. Pe baza teoremei (3.15) se obține :

Teorema 3.16. Presupunem că F este un operator diferențiabil Fréchet în punctul x^*, soluție a ecuației $Fx = o$ și că $F'(x^*)$ este o matrice pozitiv definită cu constanta 1 [relația (3.37)]. Atunci x^* este punct de atracție pentru iterația $x^{k+1} = x^k - \lambda Fx^k$ unde λ este un număr pozitiv care satisface relațiile (3.48).

Observația 3.18. In cazul $n=1$ metoda liniilor paralele are următoarea interpretare : iterația următoare x^{k+1} se obține intersectînd axa ox cu o paralelă la direcția fixă (λ) dusă prin punctul de coordonate $(x^k, f(x^k))$ (figura alăturată).



Este clar că în acest caz şirul generat de metoda liniilor paralele converge dacă f este o funcţie continuă strict crescătoare într-o vecinătate a soluţiei x^* şi dacă $\lambda < 1$. Condiţia (3.47) obţinută pentru convergenţa metodei în cazul m dimensional implică monotonia operatorului F într-o vecinătate a punctului x^*. Prin urmare teorema (3.16) reprezintă o generalizare naturală a metodei liniilor paralele de la cazul uni-dimensional.

Observaţia 3.19. Teorema (3.16) este prezentată în [41(1)] cu condiţia mai tare, F să aibă derivată Fréchet mărginită pe o sferă centrată în x^*.

Observaţia 3.20. Metoda liniilor paralele a fost considerată de numeroşi autori în spaţii finit dimensionale sau în spaţii Hilbert [72(2)], [34(1)], [46(1)], [66(1)].

3.7.3. Aplicaţie la ecuaţiile cu diferenţe corespunzătoare problemei Dirichlet eliptice slab neliniare.

Considerăm problema Dirichlet eliptică slab neliniară (2.15) şi ecuaţiile cu diferenţe standard (2.20). Vom scrie aceste ecuaţii sub o formă puţin modificată înmulţindu-le cu $-h^2$; obţinem astfel sistemul

$$Ax + \phi x = o$$

unde A este o matrice bloc tridiagonală de dimensiune nxn unde $n = m^2$ de forma

$$A = \begin{pmatrix} B & I & & & 0 \\ I & B & I & & \\ & \ddots & \ddots & \ddots & \\ & & B & I & \\ 0 & & & I & B \end{pmatrix} \qquad B = \begin{pmatrix} 4 & -1 & & & 0 \\ -1 & 4 & -1 & & \\ & \ddots & \ddots & \ddots & \\ & & & 4 & -1 \\ 0 & & & -1 & 4 \end{pmatrix}$$

iar ϕx este dat de $h^2\left[f(t_1,s_1,x_{11}),\ldots,f(t_m,s_1,x_{m1}),\ldots,f(t_1,s_m,x_{1m}),\ldots,f(t_m,s_m,x_{mm})\right]^T$.

Pentru rezolvarea numerică a sistemului aplicăm metoda (3.42). Vom presupune că funcţia f are derivată parţială în raport cu u pe $[0,1]\times[0,1]\times\mathbb{R}$ şi că $|f_u(t,s,u)| \le \beta$ pentru $(t,s,u) \in [0,1]\times[0,1]\times\mathbb{R}$.

In acest caz este clar că Φ este diferenţiabil Fréchet pe X_n şi

$$\|\Phi'(x^*)\| = \max_{i,j} h^2 \left| f_u(t_i,s_j,x^*_{ij}) \right| \le h^2\beta$$

Pe de altă parte ţinînd cont că valorile proprii ale matricei A sînt

$$\lambda_{ij} = 4\left(\sin^2\left(\frac{i\pi h}{2}\right) + \sin^2\frac{j\pi h}{2}\right), \text{ avem}$$

$$\|A\| = \max_{i,j} 4\left(\sin^2\frac{ih\pi}{2} + \sin^2\frac{jh\pi}{2}\right) < 8$$

$$\|A^{-1}\| = \max_{i,j}\left[4\left(\sin^2\frac{ih\pi}{2} + \sin^2\frac{jh\pi}{2}\right)\right]^{-1} = \left(8\sin^2\frac{h\pi}{2}\right)^{-1}$$

de unde rezultă

$$\|A^{-1}\|\,\|\Phi'(x^*)\| \le \frac{h^2}{8\sin^2\frac{h\pi}{2}} \to \frac{\beta}{2\pi^2} \quad \text{cînd} \quad h \longrightarrow \infty .$$

Prin urmare, dacă punem condiţia $\beta < 2\pi^2$, atunci pentru un n suficient de mare $\|A^{-1}\|\,\|\Phi'(x^*)\| < 1$ şi condiţia (3.43) din teorema (3.14) este satisfăcută. Să mai observăm că deoarece $\|A\| < 8$ constanta λ se poate lua $1/64$ (în acest caz $\lambda = 1/64 < \|A\|^{-2}$). Ţinînd cont şi de teorema (2.5) obţinem :

Corolarul 1. Presupunem că f are derivată parţială în raport cu u pe $[0,1]\times[0,1]\times\mathbb{R}$ şi că

$$|f_u(t,s,u)| \le \beta < 2\pi^2 \text{ pentru orice } (t,s,u) \in [0,1]\times[0,1]\times\mathbb{R}$$

atunci, pentru n suficient de mare, ecuaţia $Ax + \Phi x = 0$ are soluţie unică $x^* \in \mathbb{R}^n$ care este un punct de atracţie pentru iteraţia

$$x^{k+1} = x^k - 64^{-1}A^T(Ax^k + \Phi x^k)$$

Observaţia 3.42. După cum rezultă din testele numerice efectuate

(paragraful 4.4) rata de convergenţă a acestei metode de calcul este scăzută. Dar acest rezultat are o anumită importanţă de principiu. Teoremele (2.5) şi (2.6) asigură existenţa şi unicitatea soluţiilor ecuaţiilor cu diferenţe standard şi respectiv convergenţa acestora la soluţia problemei continue (2.13) dacă $f_u \geq -\gamma > -2\pi^2$; prin urmare, este interesant să dispunem şi de o metodă numerică de calcul care să asigure deţinerea soluţiilor ecuaţiilor cu diferenţe în aceleaşi condiţii. Metoda considerată satisface în parte acest deziderat, condiţia principală care se cere pentru convergenţă ($|f_u| \leq \beta < 2\pi^2$) fiind de aceeaşi natură cu condiţia principală din teoremele (2.5) şi (2.6).

Un rezultat similar se poate obţine şi prin aplicarea metodei liniilor paralele. Vom scrie ecuaţiile cu diferenţe sub forma $Ax + \Phi x = o$ unde A are forma (2.17) dar cu semn schimbat iar Φ este definit în 2.3.3. Cea mai mică valoare proprie a lui A este $\lambda_1 = 8h^{-2}\sin^2\frac{\pi h}{2}$ şi $\lambda_1 \longrightarrow 2\pi^2$ cînd $h \longrightarrow o$. Luăm $\overline{Fx} = Ax + \Phi x$ şi presupunem că $f_u \geq -\gamma$, obţinem

$$\langle F'(x)y,y\rangle = \langle [A + \Phi'(x)]y,y\rangle = \langle Ay,y\rangle + \langle \Phi'(x)y,y\rangle \geq (\lambda_1 - \gamma)\|y\|^2 \longrightarrow (2\pi^2 - \gamma)\|y\|^2.$$

Condiţia (3.47) se reduce la $2\pi^2 - \gamma > 1$ sau $\gamma < 2\pi^2 - 1$.

Din teorema (3.16) rezultă :

Corolarul 2. Presupunem că f are derivată parţială în raport cu u pe $[o,1]x[o,1]x\mathbb{R}$ şi că

$f_u(t,s,u) \geq -\gamma > -2\pi^2 + 1$ pentru orice $(t,s,u) \in [o,1]x[o,1]x\mathbb{R}$.

Atunci pentru n suficient de mare, ecuaţia $Ax + \Phi x = o$ are o unică soluţie $x^* \in \mathbb{R}^n$ care este punct de atracţie pentru metoda liniilor paralele

$$x^{k+1} = x^k - \lambda(Ax^k + \Phi x^k).$$

Rezultate asemănătoare se pot enunţa şi pentru ecuaţiile cu diferenţe corespunzătoare problemei la limită bilocale slab neliniare (2.2).

3.8. Metoda lui Fridman

Ţinînd cont de eficienţa deosebită a metodei lui Fridman pentru rezolvarea ecuaţiilor neliniare (paragraful 4.3) vom prezenta în cele ce urmează o analiză separată a convergenţei acestei metode în spaţii finit dimensionale [58(3),(6)]. Condiţiile de convergenţă sînt substanţial mai slabe decît cele din corolarul 3 (punctul 3.6.5) sau decît cele din lucrarea [21(1)].

3.8.1. Introducere. După cum s-a mai precizat, metoda lui Fridman (3.39) este o metodă de tip gradient în care lungimea pasului este $\lambda_k = \|Fx^k\|^2 / \|F'(x^k)^T Fx^k\|^2$. Această valoare se obţine printr-o anumită aproximare a valorii care minimizează funcţionala $\|x^k - F(x^k)^T Fx^k\|^2$. Metoda lui Fridman se poate obţine şi pe calea următoare. Pornim de la metoda lui Newton $x^{k+1} = x^k - F'(x^k)^{-1} Fx^k$ şi determinăm iteraţia următoare x^{k+1} rezolvînd sistemul liniar

$$F'(x^k)z = F'(x^k)x^k - Fx^k$$

Să notăm $F'(x^k) = A$, $b = F'(x^k)x^k - Fx^k$. Vom rezolva acest sistem cu metoda iterativă propusă de Altman [1(1)]

$$z^{j+1} = z^j - \frac{\|Az^j - b\|^2}{\|A^T(Az^j - b)\|^2} \cdot A^T(Az^j - b)$$

care converge cu singura condiţie ca A să fie nesingulară. Soluţia acestui sistem z^* este tocmai iteraţia următoare x^{k+1}. Intrucît z^* (sau x^{k+1}) este apropiat de x^k vom luc $z^0 = x^k$ şi vom lua în procesul iterativ de mai sus o singură iteraţie deci vom face aproximarea $z^* \cong z^1$. Efectuînd calculele se obţine

$$x^{k+1} = x^k - \frac{\|Fx^k\|^2}{\|F'(x^k)^T Fx^k\|^2} F'(x^k)^T \cdot Fx^k \qquad (3.49)$$

adică tocmai metoda lui Fridman (3.39).

Să observăm că în cazul $m = 1$ această metodă coincide cu metoda lui Newton.

3.8.2. Analiza convergenţei. Vom demonstra mai întîi o lemă ajutătoare. Ca şi în paragraful precedent vom considera norma euclidiană pe spaţiul $\mathbb{R}^n$.

Lema 3.11. Presupunem că F este un operator diferenţiabil Fréchet pe o sferă deschisă $S = S(x^0,r)$ şi că următoarele condiţii sînt îndeplinite

1) $\|F'(u)-F'(v)\| \leq K\|u-v\|$ pentru orice $u,v \in S$

2) există $F'(x)^{-1}$ şi $\|F'(x)^{-1}\| \leq B$ pentru orice $x \in S$

3) $rKB < 1$

Atunci

$$2 < y-x, F'(x)^T(Fy-Fx) > \geq \|Fy-Fx\|^2 \text{ pentru orice } x,y \in S \quad (3.50)$$

Demonstraţie. Ţinînd cont de condiţia 1) putem aplica formula de medie cunoscută $\|Fy-Fx-F'(x)(y-x)\| \leq \frac{1}{2}K\|y-x\|^2$ pentru orice $x,y \in S$. Dacă notăm cu $R(x,y) = Fy-Fx-F'(x)(y-x)$ şi utilizăm această relaţie şi condiţiile 2) şi 3) obţinem pentru orice $x,y \in S$

$$\|F'(x)(y-x)\| \geq \|F'(x)^{-1}\|^{-1}\|y-x\| \geq B^{-1}\|y-x\| \geq$$

$$\geq rK\|y-x\| > \frac{1}{2}K\|y-x\|^2 \geq \|R(x,y)\|$$

Aşadar

$$2 < y-x, F'(x)^T(Fy-Fx) > -\|Fy-Fx\|^2 = 2 < F'(x)(y-x), F'(x)(y-x)+$$

$$+ R(x,y) > -\|F'(x)(y-x)+R(x,y)\|^2 = \|F'(x)(y-x)\|^2 - \|R(x,y)\|^2 > 0.$$

Observaţia 3.22. Lema rămîne valabilă în spaţii Hilbert reale cu o singură modificare ca în condiţia 2) să presupunem că operatorul $F'(x)$ are invers mărginit pentru orice $x \in S$.

Observaţia 3.23. Dacă pentru $x^0 \in S$ există o vecinătate $V \subset S$ a lui x^0 astfel încît $y \in V$, $y \neq x^0$ rezultă că $Fy \neq Fx^0$, atunci din continuitatea lui F şi a lui F' rezultă că $< y-x, F'(x^0)^T(Fy-Fx) > > 0$ pe o anumită sferă centrată în x^0 şi pentru $x \neq y$. Aceasta înseamnă că operatorul $F'(x^0)^T F$ este mo-

notoп. O astfel de situaţie apare ca naturală dacă se ţine cont că metoda (3.49) poate fi considerată ca o metodă a liniilor paralele aplicată operatorului $F'(x^0)F$.

In particular, dacă x^* este o soluţie a ecuaţiei $Fx = o$ şi luăm $y = x^*$, relaţia (3.5o) devine

$$2 \langle x - x^*, F'(x)^T Fx \rangle > \| Fx \|^2 \qquad (3.51)$$

Teorema 3.17. Presupunem că toate condiţiile din lema 3.11 sînt satisfăcute şi că x^* este soluţie a ecuaţiei $Fx = o$. Atunci şirul $\{x^k\}$ generat de (3.49) converge la x^*.

Demonstraţie. Vom arăta că operatorul $T : S \setminus \{x^*\} \longrightarrow R^m$ definit de

$$Tx = x - \frac{\| Fx \|^2}{\| F'(x)^T Fx \|^2} F'(x)^T Fx$$

este cvasineexpansiv pe $S \setminus \{x^*\}$. Avem :

$$\| Tx - x^* \|^2 = \| x - x^* - \frac{\| Fx \|^2}{\| F'(x)^T Fx \|^2} F'(x)^T Fx \|^2 = \| x - x^* \| - 2 \frac{\| Fx \|^2}{\| F'(x)^T Fx \|^2} \cdot$$

$$\langle x - x^*, F'(x)^T Fx \rangle + \frac{\| Fx \|^4}{\| F'(x)^T Fx \|^2} = \| x - x^* \|^2 -$$

$$- \frac{\| Fx \|^2}{\| F'(x)^T Fx \|^2} (2 \langle x - x^*, F'(x)^T Fx \rangle - \| Fx \|^2).$$

Ţinînd cont de (3.51) rezultă

$$\| Tx - x^* \|^2 < \| x - x^* \|^2$$

prin urmare T este cvasi-neexpansiv şi deci putem aplica lema 3.3.

Observaţia 3.24. Condiţiile lemei 3.11 sînt asemănătoare cu cele din teorema lui Kantorovici [28(2)].

Observaţia 3.25. Condiţiile de convergenţă în cazul nostru sînt mai slabe decît cele impuse de Fridman [21(1)].

Scanned with OKEN Scanner

CAPITOLUL 4.

ALGORITMI DE CALCUL ŞI PROGRAME

In acest capitol se analizează algoritmii de calcul corespunzători metodelor numerice prezentate în capitolul 3 şi se dau programele FORTRAN corespunzătoare. Exemplele numerice considerate sînt ecuaţii cu diferenţe corespunzătoare unor probleme la limită neliniare concrete (ecuaţii diferenţiale ordinare sau cu derivate parţiale) selectate astfel încît să se pună în evidenţă particularităţile metodelor numerice aplicate.

4.1. Metoda iteraţiilor simple

In acest paragraf se consideră ecuaţia (3.17) şi metoda iterativă (3.18) (metoda iteraţiilor simple), pentru care se prezintă un program FORTRAN general.

4.1.1. Algoritmul de calcul. Metoda iterativă (3.18) :

$$A(x^k)x^{k+1} = \Phi x^k$$

comportă ca rezolvare principală, rezolvarea unui sistem liniar cu matricea $A(x^k)$ şi termeni liberi Φx^k, la fiecare pas de iteraţie. Algoritmul de calcul se poate concepe pentru ecuaţii neliniare de forma (3.17) oarecare, sau pentru ecuaţii particulare, de exemplu pentru ecuaţii cu diferenţe corespunzătoare unor probleme la limită neliniare. In cazul general este necesară o procedură pentru rezolvarea sistemelor liniare oarecare sau utilizarea subprogramelor corespunzătoare din biblioteca matematică ; în cazuri particulare se pot utiliza proceduri speciale în scopul reducerii consumului de timp şi de memorie.

Vom trata în continuare cazul general şi cazul ecuaţiilor cu diferenţe corespunzătoare unei probleme la limită bilocale slab neliniare.

Cazul general. Principalele etape de calcul sînt :

Etapa 1. Se atribuie lui x valoarea x^0 ; prin x notăm iteraţia curentă (x^k) iar x^0 se citeşte dintr-un fişier de intrare sau se generează cu instrucţia DATA.

Etapa 2. Se calculează matricea A şi termenul liber B utilizînd valoarea x (iteraţie curentă) ;

Etapa 3. Se calculează iteraţia următoare y prin rezolvarea sistemului liniar $Ay = B$;

Etapa 4. Se compară iteraţia actuală x cu iteraţia următoare y; dacă $\| x-y \| < \varepsilon$ procesul de calcul se opreşte ; în caz contrar se înlocuieşte x cu y şi se continuă etapa a doua.

Programul FORTRAN

```
      DIMENSION X(6),A(6,6),B(6)
      DATA X/6*Ø./
      DATA N,EPS,LK/6,Ø.ØØ1,1ØØ/
      CALL ITSIM(N,EPS,A,B,HP,X,LK,KOD)
      WRITE(1Ø8,1) KOD
1     FORMAT(1X,'KOD=',I3)
      STOP
      END
C
C     PRECIZAM SISTEMUL A(X)X=FI(X)PRIN A=A(X) SI B=FI(X)
      SUBROUTINE EC(N,A,B,HP,X)
      DIMENSION A(N,N),B(N),X(N)
      DO 1Ø I=1,N
      DO 1Ø J=1,N
1Ø    A(I,J)=Ø.
```

Scanned with OKEN Scanner

```
      A(1,1)=2+HP*SIN(X(1))
      A(1,2)=-1.
      A(N,N)=2+HP*SIN(X(N))
      A(N,N-1)=-1.
      B(1)=HP
      B(N)=HP
      L=N-1
      DO 3 I=2,L
      A(I,I-1)=-1.
      A(I,I+1)=-1.
      A(I,I)=2+HP*SIN(X(I))
3     B(I)=HP
      RETURN
      END
C
C REZOLVAREA SISTEMULUI
      SUBROUTINE SISTEM(A,B,N)
      DIMENSION A(N,N),B(N)
      EPS=1.E-8
      K1=N-1
      DO 10 K=1,K1
      AM=0
      DO 11 I=K,N
      IF(ABS(A(I,K)).LE.AM) GOTO 11
      AM=ABS(A(I,K))
      IO=I
11    CONTINUE
      IF(AM.GT.EPS) GOTO 20
22    WRITE(108,21)
21    FORMAT(5X,'SISTEM LINIAR INCOMPATIBIL')
      STOP
```

Scanned with OKEN Scanner

```
2Ø      DO 12 J=K,N
        X=A(K,J)
        A(K,J)=A(IØ,J)
12      A(IØ,J)=X
        X=B(K)
        B(K)=B(IØ)
        B(IØ)=X
        K2=K+1
        DO 13 I=K2,N
        X=A(I,K)/A(K,K)
        DO 14 J=K,N
14      A(I,J)=A(I,J)-A(K,J)*X
13      B(I)=B(I)-B(K)*X
1Ø      CONTINUE
        IF(ABS(A(N,N).LT.EPS) GOTO 22
        B(N)=B(N)/A(N,N)
        K=N-1
23      K1=K+1
        DO 15 J=K1,N
15      B(K)=B(K)-A(K,J)*B(J)
        B(K)=B(K)/A(K,K)
        K=K-1
        IF(K.GE.1) GOTO 23
        RETURN
        END
C
C METODA ITERATIILOR SIMPLE A(X)X-FI(X)=Ø
        SUBROUTINE ITSIM(N,EPS,A,B,HP,X,LK,KOD)
        DIMENSION A(N,N),B(N),X(N)
        H=1./(N+1)
        HP=H*H
```

Scanned with OKEN Scanner

```
      K=Ø
      KØD=Ø
      WRITE(1Ø8,2Ø)
2Ø    FORMAT(1X,'ITERATIA',1ØX,'SOLUTIA')
4     WRITE(1Ø8,1)(X(I),I=1,N)
      FORMAT(1X,I3,6X,(1ØF12.6))
      CALL EC(N,A,B,HP,X).
      CALL SISTEM(A,B,N)
      M=Ø
      DO 2 I=1,N
      IF(ABS(B(I)-X(I)).GE.EPS) M=1
2     CONTINUE
      IF(M.EQ.Ø) RETURN
      DO 3Ø I=1,N
3Ø    X(I)=B(I)
      K=K+1
      IF(K.LE.LK) GOTO 4
      KOD=2
      RETURN
      END
```

Pentru N=6 în două iteraţii s-a obţinut soluţia :

X=(Ø.Ø6Ø637 Ø.1ØØ942 Ø.121Ø48 Ø.121Ø48 Ø.1ØØ942 Ø.Ø6Ø637)

4.1.2. Cazul ecuaţiilor cu diferenţe (2.4). Vom înmulţi ecuaţiile (2.4) cu $-h^2$ astfel încît obţinem

$$Ax = \Phi x$$

unde matricea A este dată de (2.1), exceptînd factorul $-h^2$ şi $\Phi x = -h^2 \cdot (f(t_1,x_1),\ldots,f(t_n,x_n))^T$. Întrucît matricea A^{-1} se poate determina explicit $[5^{-1}(1)]\,[3Ø(1)]\,[12(5)]$, vom scrie procesul iterativ (3.18) sub forma :

$$x^{k+1} = A^{-1}\overline{\Phi}x^k \qquad (4.1)$$

este clar că în acest caz calculul se simplifică substanţial, rezolvarea unui sistem liniar la fiecare pas de iteraţie înlocuindu-se cu operaţia mai simplă de înmulţire a unei matrici cu un vector. Să mai observăm că întrucît valoarea numerică a unui element a lui A^{-1} depinde numai de cei doi indici ai elementului este suficient să generăm succesiv cîte o linie a matricei A^{-1}, ceea ce permite obţinerea succesivă a elementelor produsului $A^{-1}\overline{\Phi}x^k$. Se evită astfel stocarea matricei A^{-1} şi un consum mare de memorie. Principalele etape de calcul sînt :

Etapa 1. Se atribuie lui x valoarea iniţială x^0 ;

Etapa 2. Se calculează iteraţia următoare y prin înmulţirea matricei A^{-1} cu $\overline{\Phi}x$, $y = A^{-1}\overline{\Phi}x$;

Etapa 3. Se compară iteraţia curentă x cu iteraţia următoare y; dacă $\|x-y\| < \varepsilon$, procesul de calcul se opreşte ; în caz contrar se înlocuieşte x cu y şi se continuă cu etapa a doua.

Produsul dintre matricea A^{-1} şi x se efectuează fără generarea matricei A^{-1}, ţinînd cont că

$$a^{-1}_{ij} = \begin{cases} \frac{i(n-j+1)}{n+1} & i \leq j \\ \frac{j(n-i+1)}{n+1} & i \geq j \end{cases}$$

Componentele lui $\overline{\Phi}x$ se calculează cu ajutorul unei funcţii definite F cu doi parametri T şi Z , reprezentînd respectiv valorile actuale ale lui t_j şi x_j (partea dreaptă a funcţiei definite coincide cu f). In general într-o problemă la limită de forma (2.2) funcţia f este definită cu o singură formulă explicită, astfel încît utilizarea unei funcţii definite apare ca cel mai simplu şi mai natural procedeu de a calcula componentele lui $\overline{\Phi}x$; dacă f este definită în alt mod, calculul acestor componente se va efectua într-un subprogram iar în programul principal se vor prevedea instrucţiile de apel corespunzătoare. Ca şi în ca-

zul general, după calculul fiecărei iteraţii se imprimă numărul curent al iteraţiei şi componentele iteraţiei curente ; numărul de iteraţii este limitat de IK.

```
      DIMENSION Y(1ØØ),X(1ØØ)
      F(TJ,Z)=TJ-TJ+Z**3
      DATA N,EPS,ALFA,BETA,LK/6,Ø.ØØØØ1,Ø.,1.,1ØØ/
      H=1./(N+1)
      HP=H*H
      K=Ø
      T=H
      DO 1 I=1,N
      X(I)=(BETA-ALFA)*T+ALFA
1     T=T+H
7     WRITE(1Ø8,2)K,(X(I),I=1,N)
2     FORMAT(5X,'ITERATIA',I4/(5X,1ØF1Ø.6))
      K=K+1
      IF(K.GT.LK)STOP
      M=Ø
      DO 3 I=1,N
      Y(I)=Ø.
      T=H
      DO 4 J=1,N
      IF(I.LE.J) GOTO 5
      Y(I)=Y(I)+J*(N-I+1)*F(T,X(J))
      GOTO 4
5     Y(I)=Y(I)+I*(N-J+1)*F(T,X(J))
4     T=T+H
      Y(I)=-(Y(I)*HP-ALFA*(N-I+1)-BETA*I)/(N+1)
      IF(ABS(Y(I)-X(I)).LE.EPS) GOTO 3
      M=1
3     CONTINUE
```

```
      IF(M.EQ.Ø)STOP
      DO 6 J=1,N
6     X(J)=Y(J)
      GOTO 7
      END
```

După 4 iteraţii,pentru N=6 s-a obţinut soluţia :

x = (Ø.156618, Ø.273288, Ø.41Ø374, Ø.548871, Ø.69Ø741, Ø.839337)

4.1.3. Cazul ecuaţiilor cu diferenţe cu corecţii (2.5) şi (2.6)

Prezentăm în cele ce urmează un program scris în FORTRAN-77 ca o variantă la programul prezentat pentru rezolvarea ecuaţiilor cu diferenţe (2.4) dar care rezolvă ecuaţiile cu diferenţe cu corecţii de ordinul doi (2.6). Ideea rezolvării ecuaţiei (2.6) cît şi etapele de lucru nu diferă esenţial de cazul standard.

```
      PROGRAM B2
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(N=3,M=9)
      CHARACTER*1 FO
      DIMENSION FO(9)
      DATA FO(1),FO(3),FO(4),FO(5),FO(6)/
     1'(','F','1','8','.'/
      DIMENSION XN(M,N),XV(M,N),F(Ø:M+1,N),AL(N),BE(N)
      COMMON/E1/XV,F/E2/A,B,H1
      H(T,J)+((AL(J)-BE(J))*T+A*BE(J)-B*AL(J))/(A-B)
      CALL ASSIGN(1,'SY:B2.DAT')
      CALL ASSIGN(2,'SY:B2.LST')
      READ(1,1Ø1)EPS,A,B,(AL(I),I=1,N)(BE(I),I=1,N)
     1,((XV(I,J),J=1,N),I=1,M)
1Ø1   FORMAT(3F,<2*N>F/(<N>F))
      WRITE(2,1Ø2)A,B,(AL(I),I=1,N),(BE(I),I=1,N)
1Ø2   FORMAT(1X,'METODA ITERATIILOR SIMPLE',
```

```
      1'ALICATATA'/
      11X,'ECUATIILOR CU DIFERENTE FINITE CU'/
      1'CORECTII DE ORDINUL DOI'//
      11X,'CONDITIILE LA LIMITA'//
      11X,'a=',F9.5,' b=',F9.5/
      1(1X,'(',< N-1 > (F9.5,','),F9.5,')'))
      WRITE(2,103)((XV(I,J),J=1,N),I=1,M)
103   FORMAT(/1X,'SOLUTIA INITIALA'//(< N > F9.5))
      H1=1./DFLOAT(M+1)
      H6=H1*H1*H1*H1*H1*H1
      IF(EPS.LT.H6)WRITE(2,104)EPS,H6
104   FORMAT(1X,' ! EPS = ',F16.15,'< AT ',
      1' O(H**6) =',F16.15)
      H12=H1*H1/12.
      H4=H1*H1*H1*H1/240.
      IP=0
2     CALL FI
      IP=IP+1
      DO 4 I=1,M
      DO 4 J=1,N
      Q=0.
      DO 3 K=1,M
3     Q=Q+A1(I,K)*F(K,J)
4     XN(I,J)=H12*F(I,J)-Q+H4*(F(I-1,J)-2.*F(I,J)+F(I+1,J))
      EP=0.
      DO 6 I=1,M
      E=0.
      DO 5 J=1,N
5     E=E+DABS(XN(I,J)-XV(I,J))
      IF(E.GT.EP) EP=E
6     CONTINUE
```

Scanned with OKEN Scanner

```
      IF(EP.LT.EPS) GOTO 8
      DO 7 I=1,M
      DO 7 J=1,N
7     XV(I,J)=XN(I,J)
      GOTO 2
8     DO 9 I=1,M
      T=I*H1
      DO 9 J=1,N
9     XN(I,J)=XN(I,J)+H(S(T),J)
      WRITE(2,105) EPS,IP
105   FORMAT(//1X,'SOLUTIILE PROBLEMEI  CU '
     1'PRECIZIA DE:',F16.15//
     1'OBTINUTE IN',I3,'PASI'//)
      I=INT(LOG10(1./EPS)+1.)
      IF(I.LT.10)THEN
        ENCODE(1,106,F0(2))N
        ENCODE(1,106,F0(7))I
106     FORMAT(I1)
        F0(8)=')'
        F0(9)=' '
      ELSE
        ENCODE(1,106,F0(2) N
        I=I-10
        F0(7)='1'
        ENCODE(1,106,F0(8)) I
        F0(9)='/'
      ENDIF
      WRITE(2,F0) (AL(J),J=1,N),
     1((XN(I,J),J=1,N),I=1,M),(BE(J),J=1,N)
      STOP
      END
```

Scanned with OKEN Scanner

```
      DOUBLE PRECISION FUNCTION A1(I,J)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(N=3,M=9)
      IF(I.LE.J) THEN
        A1=DFLOAT(I)*DFLOAT(M-J+1)/DFLOAT(M+1)
        RETURN
      ELSE
        A1=DFLOAT(J)*DFLOAT(M-I+1)/DFLOAT(M+1)
        RETURN
      ENDIF
      END
      SUBROUTINE FI
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      PARAMETER(N=3,M=9)
      DIMENSION XV(M,N),F(0:M+1,N)
      COMMON /E1/XV,F/E2/A,B,H1
      G1(T,V1,V2,V3)=SQRT(2.)*(V1+V2+V3/2.)+DSIN(B*T)
      G2(T,V1,V2,V3)=SQRT(3.)*V1-V2-SQRT(5.)*V3/2.
     1+DSIN(B*T/2)
      G3(T,V1,V2,V3)=2.*SQRT(5.)*V1+SQRT(3.)+A*V3/2.
     1+DSIN(2.*T)
      H2=H1*H1
      DO 1 I=1,M
      F(I,1)=H2*G1(S(I*H1),XV(I,1),XV(I,2),XV(I,3))
      F(I,2)=H2*G2(S(I*H1),XV(I,1),XV(I,2),XV(I,3))
      F(I,3)=H2*G3(S(I*H1),XV(I,1),XV(I,2),XV(I,3))
1     CONTINUE
      DO 2 J=1,N
      F(0,J)=0.
2     F(M+1,J)=0.
      RETURN
```

Scanned with OKEN Scanner

```
      END
      DOUBLE PRECISION FUNCTION S(T)
      IMPLICIT DOUBLE PRECISION (A-H,O-Z)
      COMMON /E2/A,B,H1
      S=(1.-T)*A+T*B
      RETURN
      END
```

Fiind dată funcţia $f:[o,1]\times\mathbb{R}^3 \longrightarrow \mathbb{R}^3$

$$f_1(t,u)= \sqrt{2}u_1 + \sqrt{2}u_2 + \sqrt{2}u_3/2 + \sin(\pi t)$$

$$f_2(t,u)= \sqrt{3}u_1 - u_2 + \sqrt{5}u_3/2 + \sin(\pi t/2)$$

$$f_3(t,u)= 2\sqrt{3}u_1 + \sqrt{3}u_2 + \sqrt{2}u_3/2 + \sin(2t)$$

unde $u = (u_1,u_2,u_3)$. Considerăm următoarea problemă :

$$u'' = f(t,u)$$

$$u(o.)=(o.1,1.23,\ 2.345)$$

$$u(1.)=(2.41,3.75,\ 4.o9)$$

Evident că $L_f = \sqrt{2} + \sqrt{3} + 2\sqrt{5}$ care este < 7.68, deci se poate aplica metoda cu iteraţii simple pentru rezolvarea ecuaţiilor cu diferenţe finite cu corecţii de ordinul doi corespunzătoare. S-a obţinut soluţia :

```
METODA ITERATIILOR SIMPLE APLICATA
ECUATIILOR CU DIFERENTE FINITE CU
CORECTII DE ORDINUL DOI

CONDITII LA LIMITA
a=Ø.ØØØØØ b=1.ØØØØØ
( Ø.1ØØØØ, 1.23ØØØ,  2.345ØØ)
( 2.41ØØØ, 3.75ØØØ,  4.Ø9ØØØ)
SOLUTIA INITIALA
1.ØØØØØ  1.ØØØØØ  1.ØØØØØ
1.ØØØØØ -Ø.Ø1ØØØ  Ø.87ØØØ
```

1.2ØØØØ	1.12ØØØ	1.11ØØØ
Ø.8ØØØØ	Ø.78ØØØ	Ø.97ØØØ
Ø.57ØØØ	Ø.89ØØØ	Ø.76ØØØ
Ø.9ØØØØ	Ø.9ØØØØ	Ø.9ØØØØ
Ø.85ØØØ	Ø.92ØØØ	Ø.93ØØØ
-Ø.Ø12ØØ	Ø.94ØØØ	Ø.92ØØØ
Ø.99ØØØ	Ø.78ØØØ	Ø.79ØØØ

SOLUTIA PROBLEMEI CU PRECIZIA : .ØØØ1

OBTINUTA IN 11 PASI

Ø.1ØØØØØ	1.23ØØØØ	2.345ØØØ
Ø.32Ø219	1.472554	2.5ØØ145
Ø.541Ø14	1.715733	2.656631
Ø.762977	1.96Ø146	2.815757
Ø.986731	2.2Ø6382	2.978738
1.212951	2.454994	3.146679
1.442375	2.7Ø6494	3.32Ø547
1.675821	2.961347	3.5Ø1179
1.914204	3.219972	3.689249
2.158548	3.482746	3.885544
2.41ØØØØ	3.75ØØØØ	4.Ø9ØØØØ

4.1.4. Cazul ecuaţiilor cu diferenţe (2.12). Considerăm ecuaţia (2.12) sub forma :

$$A(x)x = \Phi x$$

cu matricea $A(x)$ dată de :

$$A(x)=\begin{pmatrix} 2+h^2a(x_1) & -1 & & 0 \\ -1 & 2+h^2a(x_2) & -1 & \\ & \ddots & \ddots & \ddots \\ 0 & & -1 & 2+h^2a(x_n) \end{pmatrix} \qquad (4.2)$$

şi

$$\Phi(x) = -h^2 \begin{bmatrix} f(t_1, x_1) \\ \vdots \\ f(t_n, x_n) \end{bmatrix}$$

Generarea elementelor matricei A o vom face cu ajutorul unei funcţii definite AO(X1) cu parametrul X1, reprezentînd valorile actuale ale lui x_j (partea dreaptă a funcţiei definite coincide cu a(u)). In mod asemănător componentele lui Φx le vom calcula cu funcţia definită F cu doi parametri TJ şi X, reprezentînd respectiv valorile actuale ale lui t_j şi x_j (partea dreaptă a funcţiei definite coincide cu f). In general într-o problemă la limită de forma (2.11) funcţiile f şi a(u) sînt definite cu cîte o singură formulă explicită, astfel încît utilizarea funcţiilor explicite apare ca cel mai simplu şi mai natural procedeu de a calcula componentele lui Φx şi A(x).

Rezolvarea sistemului liniar se realizează cu subprogramul SISTEM prezentat în cazul general. După calculul fiecărei iteraţii se imprimă numărul curent al iteraţiei şi componentele iteraţiei curente, numărul de iteraţii limitîndu-se la 100. Prezentăm în continuare programul FORTRAN :

```
      DIMENSION A(10,10),B(10),X(10),T(10)
      AO(X1)=SIN(X1)
      F(TJ,X1)=TJ-TJ+X1-X1-1
      DATA N,EPS,A/10,0.000001,100*0./
      H=1./(N+1)
      HP=H*H
      K=0
      DO 10 I=1,N
      X(I)=0.
10    T(I)=I*H
      WRITE(108,20)
20    FORMAT(1X,'ITERATIA',10X,'SOLUTIA')
```

```
5     WRITE(108,1) K,(X(I),I=1,N)
1     FORMAT(1X,I3,6X,(10F12.6))
      A(1,1)=2+HP*AØ(X(1))
      A(1,2)=-1
      B(1)=-HP*F(T(1),X(1))
      A(N,N)=2+HP*AØ(X(N))
      A(N,N-1)=-1
      B(N)=-HP*F(T(N),X(N))
      L=N-1
      DO 2 I=2,L
      A(I,I-1)=-1
      A(I,I+1)=-1
      A(I,I)=2+HP*AØ(X(I))
2     B(I)=-HP*F(T(I),X(I))
      CALL SISTEM(A,B,N)
      M=0
      DO 3 I=1,N
      IF(ABS(B(I)-X(I)).GE.EPS) M=1
3     CONTINUE
      IF(M.EQ.Ø) STOP
      DO 4 I=1,N
4     X(I)=B(I)
      K=K+1
      IF(K.LE.1ØØ) GOTO 5
      STOP
      END
```

Urmează subrutina SISTEM descrisă la cazul general. Iteraţiile succesive pentru diferite valori ale lui N sînt

N=1Ø

NO.	X1	X2	...	X1Ø
Ø	Ø.ØØØØØØ	Ø.ØØØØØØ	...	Ø.ØØØØØØ

1	Ø.Ø41372	Ø.Ø7438Ø	...	Ø.Ø41322
2	Ø.Ø4Ø948	Ø.Ø73646	...	Ø.Ø4Ø948
3	Ø.Ø4Ø952	Ø.Ø73654	...	Ø.Ø4Ø952

N=6

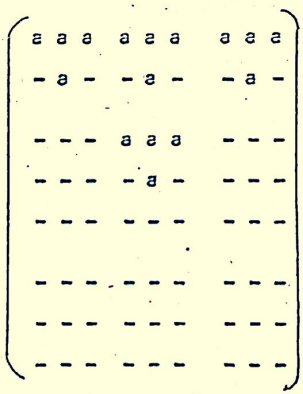
NO	X1	X2	...	X1Ø
Ø	Ø.ØØØØØØ	Ø.ØØØØØØ	...	Ø.ØØØØØØ
1	Ø.Ø61224	Ø.1Ø2Ø41	...	Ø.Ø61224
2	Ø.Ø6Ø637	Ø.1ØØ942	...	Ø.Ø6Ø637
3	Ø.Ø6Ø644	Ø.1ØØ954	...	Ø.Ø6Ø644

4.1.5. Cazul ecuaţiilor cu diferenţe cu corecţii pentru problema Dirichlet pentru ecuaţii eliptice slab neliniare.

Metoda (3.26) pentru ecuaţia (2.22) (precum şi metoda standard pentru ecuaţia (2.20)) are inconvenientul că foloseşte matricea A^{-1} dată de (2.17) care nu are proprietatea de matrice rară şi în cazul unei diviziuni cu un m mare inversarea matricei şi memorarea ei poate lua timp şi memorie mare. Folosind faptul că matricea A este matrice rară, pozitiv definită şi simetrică s-a folosit metoda factorizării Cholesky [2(1)], care este o metodă eficientă de factorizare a matricilor rare simetrice şi pozitiv definite. Obţinem matricea A ca produs de matrici. $A = U^TU$, unde U este o matrice superior triunghiulară. Inversarea matricei A revine la inversarea matricei U care din punct de vedere al memoriei şi al timpului de inversare este mult superioară inversării clasice a matricilor. Numărul de operaţii elementare este de trei ori mai mic faţă de ordinul $(m^2)^3$, operaţii necesare pentru inversare în mod clasic, iar memoria necesară este de $m^2(m^2+1)/2$ reali faţă de $2(m^2)^2$ reali necesari pentru inversarea cu metoda transformării în matricea unitate.

Pentru memorarea matricei A^{-1} observăm că această matrice va fi, ca şi matricea A, bisimetrică şi celulară, iar la rîndul

lor celulele sînt şi ele bisimetrice. ($A \in \mathcal{M}_n(\mathbb{R})$ este bisimetrică dacă $a_{ij} = a_{ji}$ şi $a_{ij} = a_{m+1-j,m+1-i}$ pentru orice $i,j=1,n$). Într-adevăr, pentru că produsul AA^{-1} este egal cu I, o matrice bisimetrică şi celulară, rezultă că A^{-1} nu poate fi decît bisimetrică şi celulară cu celule bisimetrice. Pe baza acestei observaţii este suficient să reţinem din matricea A elementele (pentru $n=3$) a_{11}^{-1}, a_{12}^{-1}, a_{13}^{-1}, a_{22}^{-1}, a_{14}^{-1}, a_{15}^{-1}, a_{16}^{-1}, a_{25}^{-1}, a_{17}^{-1}, a_{18}^{-1}, a_{19}^{-1}, a_{28}^{-1}, a_{44}^{-1}, a_{45}^{-1}, a_{46}^{-1}, a_{55}^{-1}, realizînd o compactare a matricei A^{-1} conform schemei de mai jos :



unde numărul de elemente memorate este $(n+1)/2\ (n+1-(n+1)/2)$, iar prin $[x]$ s-a notat partea întreagă a numărului x.

Programul CHOL pentru n dat ($1 \leq n \leq 15$) face factorizarea Cholesky a matricei A^{-1}, inversarea matricei superior triunghiulară U şi compactarea matricei A^{-1}, memorînd într-un fişier secvenţial elementele menţionate mai sus :

```
      PROGRAM CHOL
      PARAMETER (N21=25425)
      VIRTUAL U(N21)
      PARAMETER (NN=4096)
      DIMENSION W(NN)
      COMMON/B1/N,N2,N22
```

```
      CALL ASSIGN(1,'TI:')
      CALL ASSIGN(2,'DD:CHOL.DAT')
      WRITE(1,101)
101   FORMAT(1X,'N(<16)=',$)
      READ(1,102)N
102   FORMAT(I2)
      WRITE(2) N
      N2=N*N
      N22=2*N2
      U(1)=SQRT(A(1,1))
      DO 1 J=2,N2
1     U(J)=A(1,J)/U(1)
      DO 4 K=2,N2
      LK=L(K)
      Y=A(K,K)
      DO 2 I=1,K-1
      LI=L(I)
      X=U(LI+K)
2     Y=Y-X*X
      Y=SQRT(Y)
      U(LK+K)=Y
      DO 4 J=K+1,N2
      X=A(K,J)
      DO 3 I=1,K-1
      LI=L(I)
3     X=X-U(LI+K)*U(LI+J)
4     U(LK+J)=X/Y
      U(1)=1./U(1)
      DO 5 I=2,N2
5     U(I)=U(I)*U(1)
      DO 9 K=2,N2
```

Scanned with OKEN Scanner

```
      LK=L(K)
      DO 6 I=1,K-1
      LI=L(I)
      DO 6 J=K+1,N2
    6 U(LI+J)=U(LI+J)-U(LI+K)*U(LK+J)/U(LK+K)
      DO 7 J=K+1,N2
    7 U(LK+J)=U(LK+J)/U(LK+K)
      DO 8 J=1,K-1
      LJ=L(J)
    8 U(LJ+K)=-U(LJ+K)/U(LK+K)
    9 U(LK+K)=1./U(LK+K)
      LM=((N+1)/2)*(N+1-(N+1)/2)
      DO 11 I=1,N2
      DO 11 J=I,N2+1-I
      IM=(I-1)/N+1
      JM=(J-1)/N+1
      IF(IM+JM.GT.N+1.AND.IM.GT.JM) GOTO 11
      K=I-(IM-1)*N
      M=J-(JM-1)*N
      IF(IM+JM.GT.N+1) THEN
        IM=N+1-IM
        JM=N+1-JM
      ENDIF
      IF(IM.GT.JM) THEN
        I1=IM
        IM=JM
        JM=I1
      ENDIF
      IF(K+M.GT.N+1) THEN
        K=N+1-K
        M=N+1-M
```

Scanned with OKEN Scanner

```
      ENDIF
      IF(K.GT.M) THEN
        I1=K
        K=M
        M=I1
      ENDIF
      ND=(IM-1)*(N-IM+1)+JM-1
      L1=LM*ND
      L2=(K-1)*(N-K+1)+M
      LI=L(I)
      LJ=L(J)
      AINV=0.
      DO 1Ø I1=MAX(I,J),N2
1Ø    AINV=AINV+U(LI+I1)*U(LJ+I1)
      W(L1+L2)=AINV
11    CONTINUE
      WRITE(2) (W(I),I=1,LM*LM)
      STOP
      END
      FUNCTION A(I,J)
      COMMON /Z1/N,N2,N22
      IF(I.EQ.J) THEN
        A=4.
        RETURN
      ENDIF
      I1=(I-1)/N
      J1=(J-1)/N
      IF(IABS(I-J).EQ.1.AND.I1.EQ.J1) THEN
        A=-1.
        RETURN
      ENDIF
```

Scanned with OKEN Scanner

```
      IF(IABS(I-J).EQ.N.AND.IABS(I1-J1).EQ.1) THEN
        A=-1.
        RETURN
      ENDIF
      A=Ø.
      RETURN
      END
      FUNCTION L(I)
      COMMON/E1/N,N2,N22
      L=(I-1)*(N22-I)/2
      RETURN
      END
```

Programul B4 ce aplică metoda (3.26) pentru rezolvarea ecuației (2.22) va încărca în memorie în prealabil matricea A^{-1} compactată în fișierul secvențial CHOL.DAT. Se obține o economie substanțială de spațiu de memorie iar timpul de decompactare este rezonabil de scurt pentru că revine la calculul unor formule simple de adrese de memorie.

Programul B4 se poate adapta pentru a prelua în mod asemănător fișierul creat în programul B5 la punctul (3.3).

```
      PROGRAM B4
      PARAMETER (N=2,M=9)
      CHARACTER*5 CH
      DIMENSION XN(M*M,N),XV(M*M,N),F(M*M,N),CH(Ø:M+1)
      DIMENSION W(((M+1)/2)*(M+1-(M+1)/2)**2)
      COMMON/E1/XV,F/E2/A,B,C,D,H1,X,Y,IN/E4/W,LM
      CALL ASSIGN(1,'SY:B4.DAT')
      CALL ASSIGN(2,'SY:B4.LST')
      CALL ASSIGN(3,'SY:CHOL.DAT')
      READ(1,101) EPS,A,B,C,D,((XV(I,J),I=1,M*M),J=1,N)
```

```
101   FORMAT(5F/(<M>F))
      WRITE(2,102) A,B,C,D
102   FORMAT(1X,'PROBLEMA LUI DIRICHLET PENTRU',
      1'ECUATIA LUI POISSON'//
      11X,'A=',F9.5,'B=',F9.5,'C=',F9.5,'D=',F9.5/
      11X,'SOLUTIA INITIALA'/)
      H1=1./FLOAT(M+1)
      H4=H1*H1*H1*H1
      IF(EPS.LT.H4) WRITE(2,*)'! EPS=',EPS,
      1'< O(H**4)',H4
      DO 2 J=1,N
      DO 2 I=1,M
      S=I*H1
      X=XX(S)
      DO 1 K=1,M
      T=K*H1
      Y=YY(T)
      CALL GAMA(I,K)
      L=(I-1)*M+K
      IF(IN.EQ.0..OR.IN.EQ.1) THEN
        CH(K)='
      ELSE
        ENCODE(5,103,CH(K))XV(L,J)
103     FORMAT(F5.2)
      ENDIF
1     CONTINUE
      WRITE(2,104) (CH(L),L=1,M)
104   FORMAT(6X,<M>A5)
2     CONTINUE
      READ(3)N1
      IF(M.EQ.N1) THEN
```

Scanned with OKEN Scanner

```
      LM=((M+1)/2)*(M+1-(M+1)/2)
      READ(3) (W(I),I=1,LM*LM)
    ELSE
      WRITE(2,105) N1,M
105   FORMAT(1X,'N1 DIN CHOL.DAT =',I2,
     1'NU ESTE EGAL CU M=',I2)
      STOP
    ENDIF
    H12=H1*H1/12.
    IP=0
3   CALL FF
    IP=IP+1
    DO 5 J=1,N
    DO 5 I=1,M*M
    Q=0.
    DO 4 K=1,M*M
4   Q=Q+A1(I,K)*F(K,J)
5   XN(I,J)=H12*F(I,J)-Q
    EP=0.
    DO 7 I=1,M*M
    E=0.
    DO 6 J=1,N
6   E=E+ABS(XN(I,J)-XV(I,J))
    IF(E.GT.EP) EP=E
7   CONTINUE
    IF(EP.LT.EPS) GOTO 9
    DO 8 J=1,N
    DO 8 I=1,M*M
8   XV(I,J)=XN(I,J)
    GOTO 3
9   WRITE(2,106)
```

```
106   FORMAT(//1X,'PUNCTELE DE PE FRONTIERA'//)
      DO 11 L=1,N
      DO 11 I=Ø,M+1
      S=I*H1
      X=XX(S)
      DO 1Ø J=Ø,M+1
      T=J*H1
      Y=YY(T)
      CALL GAMA(I,J)
      IF(IN.EQ.0) THEN
        ENCODE(5,103,CH(J)) FI(L)
      ELSE
        CH(J)='
      ENDIF
1Ø    CONTINUE
      WRITE(2,1Ø7) (CH(J),J=0,M+1)
1Ø7   FORMAT(1X,<M+2>A5)
11    CONTINUE
      WRITE(2,1Ø8) EPS,IP
1Ø8   FORMAT(//1X,'SOLUTIA CU',
      1'PRECIZIA:',F7.6//
      6X,'OBTINUT IN',I3,'PASI'//)
      DO 13 J=1,N
      DO 13 I=1,M
      S=I*H1
      X=XX(S)
      DO 12 K=1,M
      T=K*H1
      Y=YY(T)
      CALL GAMA(I,K)
      L=(I-1)*M+K
```

Scanned with OKEN Scanner

```
      IF(IN.EQ.Ø..OR.IN.EQ.1) THEN
        CH(K)='      '
      ELSE
        ENCODE(5,1Ø3,CH(K)) XN(L,J)
      ENDIF
12    CONTINUE
      WRITE(2,1Ø4) (CH(L),L=1,M)
13    CONTINUE
      STOP
      END
      SUBROUTINE FF
      PARAMETER(N=2,M=9)
      DIMENSION XV(M*M,N),F(M*M,N),PS(N)
      COMMON/E1/XV,F/E2/A,B,C,D,H1,X,Y,IN/E3/PS
      F1(X,Y,U1,U2)=X+Y-U1-U2
      F2(X,Y,U1,U2)=X+Y+U1+U2
      H2=H1*H1
      DO 1 J=1,N
      DO 1 I=1,M*M
1-    F(I,J)=Ø.
      DO 2 I=1,M
      S=I*H1
      X=XX(S)
      DO 2 J=1,M
      T=J*H1
      Y=YY(T)
      L=(I-1)*M+J
      CALL GAMA(I,J)
      IF(IN.EQ.1.OR.IN.EQ.Ø) GOTO 2
      CALL PSI(I,J)
      F(L,1)=H2*F1(X,Y,XV(L,1),XV(L,2))-PS(1)
```

Scanned with OKEN Scanner

```
      F(L,2)=H2*F2(X,Y,XV(L,1),XV(L,2))-PS(2)
2     CONTINUE
      RETURN
      END
      SUBROUTINE PSI(I,J)
      PARAMETER (N=2,M=9)
      DIMENSION PS(N)
      COMMON /E2/A,B,C,D,H1,X,Y,IN/E3/PS
      DO 1 K=1,N
1     PS(K)=Ø.
      S=(I-1)*H1
      X=XX(S)
      CALL GAMA(I-1,J)
      IF(IN.EQ.-1.OR.IN.EQ.1) GOTO 3
      DO 2 K=1,N
2     PS(K)=PS(K)+FI(K)
3     S=(I+1)*H1
      X=XX(S)
      CALL GAMA(I+1,J)
      IF(IN.EQ.-1.OR.IN.EQ.1) GOTO 5
      DO 4 K=1,N
4     PS(K)=PS(K)+FI(K)
5     S=I*H1
      X=XX(S)
      T=(J-1)*H1
      Y=YY(T)
      CALL GAMA(I,J-1)
      IF(IN.EQ.-1.OR.IN.EQ.1) GOTO 7
      DO 6 K=1,N
6     PS(K)=PS(K)+FI(K)
```

Scanned with OKEN Scanner

```
7     T=(J+1)*H1
      Y=YY(T)
      CALL GAMA (I,J+1)
      IF(IN.EQ.-1.OR.IN.EQ.1) GOTO 9
      DO 8 K=1,N
8     PS(K)=PS(K)+FI(K)
9     T=J*H1
      Y=YY(T)
      RETURN
      END
      FUNCTION XX(S)
      COMMON/E2/A,B,C,D,H1,X,Y,IN
      XX=(1.-S)*A+S*B
      RETURN
      END
      FUNCTION YY(T)
      COMMON /E2/A,B,C,D,H1,X,Y,IN
      YY=(1.-T)*C+T*D
      RETURN
      END
      FUNCTION A1(I,J)
      PARAMETER (N=2,M=9)
      DIMENSION W(((M+1)/2)*(M+1-(M+1)/2)**2)
      COMMON /E4/W,LM
      IM=(I-1)/M+1
      JM=(J-1)/M+1
      K=I-(IM-1)*M
      L=J-(JM-1)*M
      IF(IM+JM.GT.M+1) THEN
        IM=M+1-IM
        JM=M+1-JM
```

Scanned with OKEN Scanner

```
      ENDIF
      IF(IM.GT.JM) THEN
        I1=IM
        IM=JM
        JM=I1
      ENDIF
      IF(K+L.GT.M+1) THEN
        K=M+1-K
        L=M+1-L
      ENDIF
      IF(K.GT.L) THEN
        I1=K
        K=L
        L=L1
      ENDIF
      ND=(IM-1)*(M-IM+1)+JM-1
      L1=LM*ND
      L2=(K-1)*(M-K+1)+L
      A1=W(L1+L2)
      RETURN
      END
      SUBROUTINE GAMA(I,J)
      COMMON /E2/A,B,C,D,H1,X,Y,IN
      GAX(S)=ABS(S-(A+B)/2.)-(B-A)/2.
      GAY(T)=ABS(T-(C+D)/2.)-(D-C)/2.
      GX=GAX(X)
      GY=GAY(Y)
      IF(GX.GT.Ø..OR.GY.GT.Ø.) THEN
        IN=1
        RETURN
      ELSE
```

Scanned with OKEN Scanner

```
      H4=H1*H1*H1*H1
        IF(GX.EQ.Ø..OR.GY.EQ.Ø.) THEN
          IN=Ø
          RETURN
        ELSE
          IN=-1
          RETURN
        ENDIF
      ENDIF
      FUNCTION FI(K)
      COMMON /E2/A,B,C,D,H1,X,Y,IN
      FI1(X,Y)=X+Y+SIN(X)+COS(Y)
      FI2(X,Y)=X+Y-SIN(X)-COS(Y)
      IF(K.EQ.1) FI=FI1(X,Y)
      IF(K.EQ.2) FI=FI2(X,Y)
      RETURN
      END
```

Exemplu

Fie $D=[o,1]x[o,1]$, $f: D \times \mathbb{R}^2 \longrightarrow \mathbb{R}^2$, $u: D \longrightarrow \mathbb{R}^2$, $w: \partial D \to \mathbb{R}^2$, $f=(f_1,f_2)$, $u=(u_1,u_2)$, $f_1: D \times \mathbb{R}^2 \longrightarrow \mathbb{R}$, $f_2: D \times \mathbb{R}^2 \longrightarrow \mathbb{R}$, $u_1: D \to \mathbb{R}$, $u_2: D \longrightarrow R$, $w=(w_1,w_2)$, $w_1: \partial D \longrightarrow \mathbb{R}$, $w_2: \partial D \longrightarrow \mathbb{R}$.

$$f_1(x,y,(u_1,u_2)) = x+y-(u_1+u_2)$$
$$f_2(x,y,(u_1,u_2)) = x+y+(u_1+u_2)$$
$$w_1(x,y) = x+y + \sin(x)+\cos(y)$$
$$w_2(x,y) = x+y - \sin(x)-\cos(y)$$

$$\begin{cases} \Delta u = f(x,y,u) \\ u\,|_{\partial D} = w \end{cases}$$

PROBLEMA LUI DIRICHLET PENTRU ECUAȚIA LUI POISSON

A = 0.00000 B = 1.00000 C = 0.00000 D = 1.00000

Scanned with OKEN Scanner

SOLUTIA INITIALA

1.ØØ	1.ØØ	...	1.ØØ
1.ØØ	1.ØØ	...	1.ØØ
⋮	⋮		⋮
1.ØØ	1.ØØ	...	1.ØØ

1.ØØ	1.ØØ	...	1.ØØ
1.ØØ	1.ØØ	...	1.ØØ
⋮	⋮		⋮
1.ØØ	1.ØØ	...	1.ØØ

PUNCTELE DE PE FRONTIERA

1.ØØ	1.Ø9	...	1.54
1.19			1.74
⋮			⋮
2.84	2.93	...	3.38
-1.ØØ	-Ø.89	...	Ø.45
-Ø.99			Ø.45
⋮			⋮
-Ø.84	-Ø.73	...	Ø.61

SOLUTIA CU PRECIZIA DE Ø.Ø1

OBTINUTA IN 3 PASI

1.28	1.36	1.43	...	1.71
1.47	1.55	1.62	...	1.91
⋮	⋮	⋮		⋮
2.76	2.84	2.92	...	3.19
-Ø.89	-Ø.79	-Ø.67	...	Ø.25
-Ø.90	-Ø.80	-Ø.69	...	Ø.23
⋮	⋮	⋮		⋮
-Ø.79	-Ø.69	-Ø.57	...	Ø.35

Scanned with OKEN Scanner

4.2. Metoda direcţiilor alternate

In acest paragraf se consideră ecuaţia (3.27) şi metoda iterativă (3.28) sub forma (3.29) pentru care se prezintă un program FORTRAN general. Exemplele tratate la punctele 4.2.2 şi 4.2.3 reprezintă ecuaţiile cu diferenţe (2.4) corespunzătoare unei probleme la limită bilocală slab neliniară de forma (2.2), respectiv (2.12) corespunzătoare problemei la limită bilocală cvasiliniară (2.11).

4.2.1. Algoritmul de calcul. Metoda iterativă

$$rx^{k+1/2} + A(x^k)x^{k+1/2} = rx^k - \Phi x^k$$
$$rx^{k+1} + \Phi x^{k+1} = rx^{k+1/2} - A(x^k)x^{k+1/2}$$

comportă pentru rezolvarea primei ecuaţii, rezolvarea unui sistem liniar cu matricea $rI + A(x^k)$ şi termenii liberi $rx^k - \Phi x^k$, iar pentru cea de-a doua ecuaţie rezolvarea unui sistem neliniar, la fiecare pas de iteraţie. Pentru ecuaţiile neliniare de forma (3.29) carecare, algoritmul de calcul necesită o procedură specială SISTEM (§ 4.1.2) sau utilizarea subprogramelor corespunzătoare din biblioteca matematică pentru rezolvarea sistemului liniar. Cea de-a doua ecuaţie (3.29) scrisă sub forma :

$$rx^{k+1} + \Phi x^{k+1} - T = 0 \qquad \text{unde}$$
$$T = \left[r - A(x^k) \right] x^{k+1/2}$$

comportă rezolvarea a n ecuaţii cu o singură necunoscută

$$rx_i^{k+1} + \Phi_i x_i^{k+1} - T_i = 0 \quad i = \overline{1,n} \qquad (4.4)$$

pentru care putem aplica una din metodele cunoscute (Newton, coardei, secantei etc.). Vom trata în continuare cazul general, cazul ecuaţiilor cu diferenţe corespunzătoare problemei (2.2) şi (2.8). Cazul general. Dacă $\Phi_i \in C^2(R)$ şi $\partial\Phi_i / \partial x_i \neq 0$ $i = \overline{1,n}$ cele n ecuaţii (4.4) pot fi rezolvate folosind metoda secantei, por-

nînd cu valoarea inițială x_i^k (x_i^{k+1} se va afla în apropierea acesteia). Principalele etape de calcul sînt următoarele :

Etapa 1. Se atribuie lui X, iterația curentă (x^k), valoarea x^0;

Etapa 2. Se generează matricea $A(x^k)$ și ϕx^k utilizînd valoarea X ;

Etapa 3. Se rezolvă sistemul liniar $A_1 x^{k+1/2} = B$ cu $A_1 = rI + A(x^k)$ și $B = rx^k - \phi x^k$ utilizînd subrutina SISTEM (§ 4.1.2). In B vom obține $x^{k+1/2}$.

Etapa 4. Se rezolvă cu metoda secantei fiecare din cele n ecuații (4.4) pentru a obține x_i^{k+1} $i = \overline{1,n}$. Salvînd în prealabil pe X în Y, noua iterație x^{k+1} va fi păstrată tot în X.

Etapa 5. Se compară vechea iterație Y cu noua iterație X ; dacă $\| X-Y \| < \varepsilon$, procesul de calcul se oprește ; în caz contrar se continuă cu etapa a doua.

4.2.2. Cazul ecuațiilor cu diferențe (2.4) Vom considera ecuația sub forma $Ax + \phi x = o$ unde

$$A = \begin{pmatrix} 2 & -1 & & & \\ -1 & 2 & -1 & & 0 \\ & \ddots & \ddots & \ddots & \\ & & -1 & 2 & -1 \\ & & & -1 & 2 \end{pmatrix} \qquad \phi x = h^2 \begin{pmatrix} f(t_1,x_1) - \alpha/h^2 \\ f(t_2,x_2) \\ \vdots \\ f(t_{n-1},x_{n-1}) \\ f(t_n,x_n) - \beta/h^2 \end{pmatrix} \tag{4.5}$$

A este Lipschitz continuu și strict monoton iar ϕ este continuu și monoton [41(1)]. Vom rezolva sistemul cu metoda direcțiilor alternante după algoritmul descris la cazul general.

4.2.3. Cazul ecuațiilor cu diferențe (2.12). Considerăm ecuația (2.12) sub forma

$$A(x)x + \phi x = 0 \quad \text{cu}$$

$$A(x)=\begin{pmatrix} 2+h^2.a(x_1) & -1 & & 0 \\ -1 & 2+h^2.a(x_2) & -1 & \\ & \ddots & \ddots & \ddots \\ & & \ddots & -1 \\ 0 & & -1 & 2+h^2.a(x_n) \end{pmatrix} \qquad (4.6)$$

$$\Phi x = h^2 \begin{pmatrix} f(t_1,x_1) \\ \vdots \\ f(t_n,x_n) \end{pmatrix} \qquad (4.7)$$

Vom face ipoteza că $a(x)$ este Lipschitz continuu şi $a(x)\geq 0$ pentru $x\in\mathbb{R}$. Avem $A(x) = A_1 + h^2A_2(x)$ cu A_1 şi A_2 precizate în § 3.4.2. Vom aplica metoda direcţiilor alternante (3.29). Pentru convergenţa metodei este suficient să arătăm că $A(x)$ este continuu Lipschitz şi strict monoton, iar Φ continuu şi monoton ; atunci existenţa soluţiei x^* a ecuaţiei de mai sus, existenţa şirului $x^{1/2},x^1,\ldots$ şi convergenţa acestuia la x^* rezultă din teorema 3.12.

Faptul că $A(x)$ este continuu Lipschitz este evident, iar monotonia strictă a lui $A(x)$ rezultă din faptul că $A(x)$ este o matrice pozitiv definită. Într-adevăr, întrucît cea mai mică valoare proprie a lui A_1 este $\lambda_1 = 2-2\cos(\pi/(n+1))$ cea mai mică valoare proprie a lui $A(x)$ este

$$\inf_{\|x\|=1} \langle A(x)x,x\rangle \geq \lambda_1 + \frac{1}{(n+1)^2}(x_1^2.a(x_1)+\ldots+x_n^2.a(x_n)) \geq \lambda_1 .$$

Atunci

$$\langle A(x)x-A(x)y,x-y\rangle = \langle A(x)(x-y),x-y\rangle \geq \lambda_1 \|x-y\|^2$$

Să presupunem că f este continuu diferenţiabilă pe $[0,1]\times\mathbb{R}$ şi că $f_u(t,u)\geq 0$ pentru orice $(t,u)\in[0,1]\times\mathbb{R}$. Atunci este evident că Φ este funcţie continuă pe $\mathbb{R}^n$. Derivata Fréchet a lui Φ este dată de :

$$\Phi'(x) = \begin{pmatrix} h^2 f x_1(t_1, x_1) & & 0 \\ & \ddots & \\ 0 & & h^2 f x_n(t_n, x_n) \end{pmatrix}$$

şi este evident o matrice semipozitiv definită. Prin urmare

$$\langle \Phi y - \Phi x , y-x \rangle = \langle \int_0^1 \Phi'(x + t(y-x))(y-x)dt, y-x \rangle =$$

$$= \int_0^1 \langle \Phi'(x + t(y-x))(y-x), y-x \rangle dt \geq 0$$

adică Φ este monoton.

Pentru rezolvarea numerică, folosim algoritmul descris la cazul general.

4.2.4. Programul FORTRAN

Cazul general. Componentele lui Φx le vom calcula cu subprogramul de tip FUNCTION F cu parametrii formali X, I, N, T, HP reprezentînd :

X = componenta i a iteraţiei curente ;

I = ordinul componentei ;

N = ordinul sistemului ; dacă sistemul se obţine prin discretizarea unor ecuaţii diferenţiale (2.4, 2.12) va preciza şi pasul de discretizare H = 1./(N+1)

T = t_i cînd programul este folosit la rezolvarea sistemelor obţinute prin discretizarea unor ecuaţii diferenţiale.

HP = 1./(N+1)^2 = H * H.

Generarea matricei $A(x^k)$ se face cu subrutina MATR de parametri formali A,X,N,HP. In A vom avea matricea (de tipul nxn) pentru iteraţia curentă X.

Procedura MDA descrie metoda direcţiilor alternate care poate fi stocată într-o bibliotecă de subprograme sursă. Utilizatorul va apela această subrutină generînd în programul princi-

pal prin DATA : N, EPS (precizia de calcul), r (notat prin R), α şi β dacă sistemul se obţine prin discretizarea unei probleme la limită bilocală şi LK - numărul maxim de iteraţii. Prin subprogramele MATR şi F va preciza matricile A(x) respectiv ϕ x . Subrutina MDA va furniza codul de funcţionare al programului :

KOD = 0 SOLUTIE NORMALA

KOD = 1 DEPASIREA NUMARULUI DE ITERATII

După fiecare iteraţie se imprimă numărul curent al iteraţiei şi componentele acesteia. Prezentăm programul FORTRAN corespunzător:

```
      DIMENSION X(6),B(6),A(6,6),A1(6,6),T(6)
      DATA N,EPS,R,ALFA,BETA,LK/6,Ø.ØØØ1,2.,Ø.,Ø.,3ØØ/
      CALL MDA(N,EPS,X,R,ALFA,BETA,LK,KOD,A,B,A1,T)
      PRINT 1,KOD
1     FORMAT(1ØX,'KOD=',I1)
      STOP
      END
C METODA DIRECTIILOR ALTERNATE
      SUBROUTINE MDA(N,EPS,X,R,ALFA,BETA,LK,KOD,A,B,A1,T)
      DIMENSION X(N),B(N),A(N,N),A1(N,N),T(N)
      K=Ø
      KOD=Ø
      H=1./(M+1)
      HP=H*H
      DO 2Ø I=1,N
      T(I)=ALFA+I*H
2Ø    X(I)=Ø.
      PRINT 2
2     FORMAT(1X,'ITERATIA',1ØX,'SOLUTIA')
1     PRINT 3,K,(X(I),I=1,N)
3     FORMAT(1X,I3,6X,(1ØF12.6))
```

```
      CALL MATR(A,X,N,HP)
      DO 4 I=1,N
      DO 4 J=1,N
      Y=A(I,J)
      IF(I.EQ.J) Y=Y+R
4     A1(I,J)=Y
      DO 5 I=1,N
5     B(I)=R*X(I)-F(X(I),I,N,T(I),HP)
      CALL SISTEM(A1,B,N)
      M=Ø
      H1=Ø.1
      DO 6 I=1,N
      H2=H1
      T1=R*B(I)
      DO 7 J=1,N
7     T1=T1-A(I,J)*B(J)
      Y=X(I)
      VF=F(X(I),I,N,T(I),HP)
      VF=VF+R*X(I)-T1
      X(I)=X(I)+H2
11    VFR=F(X(I),I,N,T(I),HP)+R*X(I)-T1
      Z=X(I)
      X(I)=X(I)-H2-H2*VF/(VFR-VF)
      IF(ABS(VFR).LT.EPS) GOTO 1Ø
      VF=VFR
      H2=X(I)-Z
      GOTO 11
1Ø    IF(ABS(X(I)-Y).GT.EPS) M=1
6     CONTINUE
      IF(M.EQ.Ø) STOP
      K=K+1
```

Scanned with OKEN Scanner

```
      IF(K.LE.LK) GOTO 1
      KOD=1
      RETURN
      END
C PRECIZAREA FUNCTIEI ϕ(X)
      FUNCTION F(X,I,N,T,HP)
      F=-HP
      RETURN
      END
C MATRICEA A(X)
      SUBROUTINE MATR(A,X,N,HP)
      DIMENSION A(N,N),X(N)
      DO 1 I=1,N
      DO 1 J=1,N
1     A(I,J)=Ø.
      A(1,1)=2+HP*SIN(X(1))
      A(N,N)=2+HP*SIN(X(N))
      A(1,2)=-1
      A(N,N-1)=-1
      L=N-1
      DO 2 I=2,L
      A(I,I)=2+HP*SIN(X(I))
      A(I,I-1)=-1
2     A(I,I+1)=-1
      RETURN
      END
C REZOLVAREA SISTEMULUI LINIAR
      SUBROUTINE SISTEM(A,B,N)
      DIMENSION A(N,N),B(N)
      EPS=1.E-8
      K1=N-1
```

Scanned with OKEN Scanner

```
      DO 10 K=1,N1
      AM=0.
      DO 11 I=K,N
      IF(ABS(A(I,K)).LE.AM) GOTO 11
      AM=ABS(A(I,K))
      IO=I
11    CONTINUE
      IF(AM.GT.EPS) GOTO 20
22    WRITE(108,21)
21    FORMAT(15X,'SISTEM LINIAR INCOMPATIBIL')
      STOP
20    DO 12 J=K,N
      X=A(K,J)
      A(K,J)=A(IO,J)
12    A(IO,J)=X
      X=B(K)
      B(K)=B(IO)
      B(IO)=X
      K2=K+1
      DO 13 I=K2,N
      X=A(I,K)/A(K,K)
      DO 14 J=K,N)
14    A(I,J)=A(I,J)-A(K,J)*X
13    B(I)=B(I)-B(K)*X
10    CONTINUE
      IF(ABS(A(N,N).LT.EPS) GOTO 22
      B(N)=B(N)/A(N,N)
      K=N-I
23    K1=K+1
      DO 15 J=K1,N
15    B(K)=B(K)-A(K,J)*B(J)
```

Scanned with OKEN Scanner

```
      B(K)=B(K)/A(K,K)
      K=K-I
      IF(K.GE.I) GOTO 23
      RETURN
      END
```

Am considerat ecuaţia $A(x)x + \Phi x = o$ cu

$$A(x) = \begin{pmatrix} 2+\sin x_1 & -1 & & 0 \\ -1 & 2+\sin x_2 & -1 & \\ & \ddots & \ddots & \ddots \\ & & \ddots & -1 \\ 0 & & -1 & 2+\sin x_n \end{pmatrix} \qquad x = h^2 \begin{pmatrix} -1 \\ -1 \\ \cdot \\ \cdot \\ \cdot \\ -1 \end{pmatrix}$$

Pentru $n=6$, $h^2 = 1./(n+1)^2$, după 27 iteraţii s-e obţinut soluţia:

x=(Ø.Ø6Ø412, Ø.1ØØ537, Ø.12Ø544, Ø.12Ø544, Ø.1ØØ537, Ø.Ø6Ø412)

4.2.5. Cazul ecuaţiilor cu diferenţe (2.4). Fie ecuaţia

$$\begin{cases} u'' = u^3 & u \in R \\ u(o) = o \quad u(1)=1 \end{cases}$$

A şi Φ (x) vor fi cei precizaţi de (4.5) unde $f(t,u) = u^3$. Pentru compararea rezultatelor obţinute prin această metodă, cu cele obţinute prin metoda iteraţiilor simple, am luat, în subrutina MDA prezentată în cazul general, iteraţia iniţială - eticheta 20 din această subrutină, va deveni :

```
      DO 2Ø I=1,N
      T(I)=ALFA+I*H
2Ø    X(I)=(BETA-ALFA)*T(I)+ALFA
```

Cu EPS=0.CCO1, r=2, n=6, α=o, β=1, A şi Φ x cei precizaţi mai sus, structura programului este următoarea :

```
      DIMENSION X(6),B(6),A(6,6),A1(6,6),T(6)
      DATA N,EPS,R,ALFA,BETA,LK/6,Ø.ØØØ1,2.,Ø.,1.,3ØØ/
      CALL MDA(N,EPS,X,R,ALFA,BETA,LK,KOD,A,B,A1,T)
```

```
      PRINT 1,KOD
1     FORMAT(1ØX,'KOD=',I1)
      STOP
      END
C PRECIZAM FUNCTIA F(T,X)
      FUNCTION F(X,I,N,T,HP)
      IF(I.EQ.N) GOTO 1
      F=HP*X**3
      RETURN
1     F=HP*X**3-1
      RETURN
      END
C MATRICEA A
      SUBROUTINE MATR(A,X,N,HP)
      DIMENSION A(N,N),X(N)
      DO 1 I=1,N
      DO 1 J=1,N
1     A(I,J)=Ø.
      A(1,1)=2.
      A(N,N)=2.
      A(1,2)=-1.
      A(N,N-1)=-1.
      L=N-1
      DO 2 I=2,L
      A(I,I)=2
      A(I,I-1)=-1
2     A(I,I+1)=-1
      RETURN
      END
C METODA DIRECTIILOR ALTERNATE
      SUBROUTINE MDA(N,EPS,X,R,ALFA,BETA,LK,KOD,A,B,AL,T)
      .
      .
      .
```

Scanned with OKEN Scanner

```
      RETURN
      END
C REZOLVAREA SISTEMULUI LINIAR
      SUBROUTINE SISTEM(A,B,N)
      .
      .
      .
      RETURN
      END
```

4.2.6. Cazul ecuaţiilor cu diferenţe (2.12)

Algoritmul de calcul pentru rezolvarea ecuaţiei cu diferenţe (2.12) corespunzător ecuaţiei (2.11) scrisă sub forma $A(x).x + \Phi x = o$ cu $A(x)$ şi Φx precizaţi în (4.6) respectiv (4.7) pentru $a(x) = \sin^2 x$ şi $f(t,x) = -1$, luînd pentru EPS = 0.0001, N=6, R=2, ALFA=BETA=0., determină următorul program FORTRAN :

```
      DIMENSION X(6),B(6),A(6,6),A1(6,6),T(6)
      DATA N,EPS,R,ALFA,BETA,LK/6,Ø.ØØØ1,2.,Ø.,Ø.,3ØØ/
      CALL MDA(N,EPS,X,R,ALFA,BETA,LK,KOD,A,B,A1,T)
      PRINT 1,KOD
1     FORMAT(1ØX,'KOD=',I1)
      STOP
      END
C FUNCTIA F(T,X)
      FUNCTION F(X,I,N,T,HP)
      F=-HP
      RETURN
      END
C MATRICEA A(X)
      SUBROUTINE MATR(A,X,N,HP)
      DIMENSION A(N,N),X(N)
      DO 1 I=1,N
      DO 1 J=1,N
1     A(I,J)=Ø.
```

```
      A(1,1)=2+HP*SIN(X(1))**2
      A(N,N)=2+HP*SIN(X(N))**2
      A(1,2)=-1.
      A(N,N-1)=-1.
      L=N-1
      DO 2 I=2,L
      A(I,I)=2.+HP*SIN(X(I))**2
      A(I,I-1)=-1.
2     A(I,I+1)=-1
      RETURN
      END
C METODA DIRECTIILOR ALTERNATE
      SUBROUTINE MDA(N,EPS,X,R,ALFA,BETA,LK,KOD,A,B,A1,T)
      .
      .
      .
      RETURN
      END
C REZOLVAREA SISTEMULUI LINIAR
      SUBROUTINE SISTEM(A,B,N)
      .
      .
      .
      RETURN
      END
```

In 28 iteraţii am obţinut soluţia :

x =(Ø.Ø6Ø951, Ø.1Ø1540, Ø.121817, Ø.121817, Ø.1Ø1540, Ø.Ø6Ø951)

4.3. Metoda lui Fridman.

In acest paragraf vom prezenta algoritmul de calcul pentru metoda (3.49) şi programul FORTRAN corespunzător. Subrutina ce descrie metoda lui Fridman este astfel organizată încît ea poate fi stocată într-o bibliotecă sursă. Cel ce doreşte să utilizeze metoda lui Fridman (pentru rezolvarea sistemelor neliniare) va trebui să precizeze sistemul Fx = o, derivata Fréchet a lui F şi să apeleze subrutina FRIDMAN [58(8)].

Scanned with OKEN Scanner

4.3.1. Algoritmul de calcul. Considerăm ecuaţia $Fx = o$ cu $F: D \subset \mathbb{R}^n \to \mathbb{R}^n$, F operator neliniar, diferenţiabil Fréchet pe D. Vom utiliza norma euclidiană $\|\cdot\|_2$ pe spaţiul $\mathbb{R}^n$ şi norma aplicaţie liniară corespunzătoare pe $L(\mathbb{R}^n)$. In condiţiile teoremei 3.17 (în care evident se consideră norma euclidiană) şirul (3.49)

$$x^{k+1} = x^k - \frac{\|Fx^k\|^2}{\|F'(x^k)Fx^k\|^2} \cdot F'(x^k)^T Fx^k$$

converge la soluţia x^* a ecuaţiei $Fx = o$. Am notat cu A^T transpusa unei matrici A. Distingem următoarele etape principale de calcul.

Etapa 1. Se atribuie lui X - iteraţia curentă x^k, valoarea iteraţiei iniţiale x^o.

Etapa 2. Se calculează $\|Fx^k\|^2$. Dacă $\|Fx^k\|^2 < \varepsilon$ procesul de calcul se opreşte ; în caz contrar se continuă cu etapa 3.

Etapa 3. Se calculează $F'(x^k)^T Fx^k$;

Etapa 4. Se calculează $\|F'(x^k)^T Fx\|^2$;

Etapa 5. Dacă la etapa 4 se obţine zero, sau dacă se depăşeşte numărul de iteraţii (propus) procesul de calcul se opreşte ; în caz contrar se calculează iteraţia următoare tot în X. Se reiau operaţiile cu etapa 2.

4.3.2. Programul FORTRAN

La fiecare pas de iteraţie cu metoda (3.49) avem nevoie de diferenţiala Fréchet $F'(x^k) \in L(\mathbb{R}^n)$. Dacă notăm cu $f_1, f_2, \ldots, f_n$ componentele lui F atunci :

$$F'(x) = \begin{pmatrix} \frac{\partial f_1}{\partial x_1} & \cdots & \frac{\partial f_1}{\partial x_n} \\ \vdots & & \vdots \\ \frac{\partial f_n}{\partial x_1} & \cdots & \frac{\partial f_n}{\partial x_n} \end{pmatrix}$$

Scanned with OKEN Scanner

Calculul fiecărui element al derivatei Fréchet îl vom efectua cu subprogramul FUNCTION DIF de parametri formali X,I,J,N ;

X = tablou de dimensiune n în care vom avea iterația curentă x^k;

I = indicele de linie al elementului derivatei Fréchet calculată pentru X ;

J = indicele de coloană al aceluiași element ;

N = dimensiunea n a sistemului.

Calculul valorii funcției f_i îl vom realiza cu subprogramul FUNCTION F de parametri X,I,N a căror semnificație am prezentat-o mai sus cu precizarea că aici I specifică rangul componentei funcției F. După apel în F vom avea valoarea funcției f_i pentru necunoscutele x_i $I = \overline{1,n}$. Prima instrucție executabilă a subprogramului este un GOTO calculat de forma GOTO(1,2,..,N)I, unde 1,2,...,N sînt etichete în subprogram corespunzătoare funcțiilor $f_1, f_2, \ldots, f_N$.

Produsul $F'(x^k)^T F x^k$ îl vom obține în PR, tablou de dimensiune N al subrutinei FRIDMAN. Această subrutină de parametri formali N, EPS, LK, X, KOD, PR descrie metoda lui FRIDMAN. Terminarea programului este condiționată de îndeplinirea condiției $\| Fx^k \|^2 \leq$ EPS, EPS fiind o constantă a programului, în exemplul considerat EPS = 1.E-4. Subrutina furnizează prin parametrul KOD, codul de funcționare al programului.

KOD = 0 SOLUTIE NORMALA

KOD = 1 IMPARTIRE CU ZERO (S1=0)

KOD = 2 DEPASIREA NUMARULUI DE ITERATII LK

Subrutina FRIDMAN poate fi stocată într-o bibliotecă sursă. Utilizatorul va preciza în programul principal dimensiunea N a sistemului, a produsului PR - tot N, va genera prin DATA pe lîngă N, EPS și iterația inițială x^0. Prin F și DIF va stabili funcția, respectiv derivata Fréchet a funcției F. Am considerat ca exemplu ecuația

$$Fx = \emptyset \quad \text{cu} \quad Fx = \begin{pmatrix} x_1 \\ \cos x_1 + x_2 \\ \vdots \\ \cos x_9 + x_{10} \end{pmatrix}$$

F'(x) va fi :

$$F'(x) = \begin{pmatrix} 1 & & & & 0 \\ -\sin x_1 & 1 & & & \\ & -\sin x_2 & 1 & & \\ & & \ddots & \ddots & \\ 0 & & & -\sin x_9 & 1 \end{pmatrix}$$

```
C METODA LUI FRIDMAN PENTRU SISTEME NELINIARE, VARIANTA 1.
      DIMENSION X(4),PR(4)
      DATA N,EPS,LK,X/4,1.E-4,1ØØ,Ø.5,-1.2,Ø.3,Ø.8/
      CALL FRIDMAN(N,EPS,LK,X,KOD,PR)
      PRINT 1,KOD
1     FORMAT(1ØX,'KOD=',I1)
      STOP
      END
C PRECIZAREA ECUATIEI F(X)=Ø
      FUNCTION F(X,I,N)
      DIMENSION X(N)
      GOTO(1,2,3,4)I
1     F=X(1)*X(1)-1.
      RETURN
2     F=X(1)*X(3)+1.
      RETURN
3     F=X(1)+X(2)+SIN(X(3))
      RETURN
4     F=COS(X(3))-X(4)**2
      RETURN
      END
```

```
C   DERIVATA FRECHET
      FUNCTION DIF(X,I,J,N)
      DIMENSION X(N)
      GOTO(1,2,3,4),I
1     GOTO(11,12,13,14),J
2     GOTO(21,22,23,24),J
3     GOTO(31,32,33,34),J
4     GOTO(41,42,43,44),J
11    DIF=2*X(1)
      RETURN
12    DIF=0.
      RETURN
13    DIF=0.
      RETURN
14    DIF=0.
      RETURN
21    DIF=X(3)
      RETURN
22    DIF=0.
      RETURN
23    DIF=X(1)
      RETURN
24    DIF=0.
      RETURN
31    DIF=1.
      RETURN
32    DIF=1.
      RETURN
33    DIF=COS(X(3))
      RETURN
34    DIF=0.
```

Scanned with OKEN Scanner

```
      RETURN
41    DIF=Ø.
      RETURN
42    DIF=Ø.
      RETURN
43    DIF=-SIN(X(3))
      RETURN
44    DIF=-2*X(4)
      RETURN
      END
C METODA LUI FRIDMAN
      SUBROUTINE FRIDMAN(N,EPS,LK,X,KOD,PR)
      DIMENSION X(N),PR(N)
      PRINT 2
2     FORMAT(1X,'ITERATIA',1ØX,'SOLUTIA')
1     PRINT 3,K,(X(I),I=1,N)
3     FORMAT(1X,I2,6X,(1ØF12.6))
      S=Ø.
      DO 7 I=1,N
7     S=S+F(X,I,N)*F(X,I,N)
      IF(S.LE.EPS)RETURN
      DO 6 I=1,N
      PR(I)=Ø.
      DO 6 J=1,N
6     PR(I)=PR(I)+DIF(X,J,I,N)*F(X,J,N)
      S1=Ø.
      DO 8 J=1,N
8     S1=S1+PR(I)*PR(I)
      IF(S1.EQ.Ø.) KOD=1
      IF(K.GE.LK) KOD=2
      IF(KOD.EQ.Ø.OR.KOD.EQ.2) RETURN
```

```
      K=K+1
      DO 9 I=1,N
9     X(I)=X(I)-S/S1*PR(I)
      GOTO 1
      END
```

Varianta 1, clasică, pentru metoda lui Fridman, prezentată mai sus presupune calculul valorii lui $f_i(x)$ și al $\partial f_i / \partial x_j$ în momentul în care se solicită valoarea acestora pentru x^k. Deci nu se generează în memoria centrală valorile tuturor componentelor lui F și ale tuturor componentelor derivatei Fréchet în x^k. Totuși numărul mare de instrucțiuni folosit pentru descrierea (precizarea) lor, reduce în parte acest avantaj, întrucît este descris calculul și pentru componentele nule ale derivatei Fréchet.

In cazul matricelor rare de dimensiuni mari, scheme de memorare standard și chiar tehnica prezentată în Varianta 1, devin ineficiente. In acest caz este indicat să se memoreze numai elementele nenule, salvînd astfel memorie și timp de calcul. Intrucît zona de memorare a matricei nu va fi identică cu matricea originală, este necesar ca tehnicile de memorare să încorporeze pe lîngă elementele nenule și mijloace de identificare a pozițiilor acestor elemente în matrice.

Se cunosc mai multe tehnici de memorare a matricelor rare [2(1)]. Pentru metoda lui Fridman și metoda Gradientului : (§ 4.3.2) cînd derivata Fréchet este o matrice rară am folosit memorarea compact aleatoare [58(8)]. Memorarea compact aleatoare constă în utilizarea unei zone primare, VALA, conținînd numai elementele nenule ale matricei și două zone secundare conținînd indicii de linie, ILIN și de coloană, ICOL, și elementele nenule. Deoarece fiecare element al matricei, nenul, este identificat individual, este posibil ca matricea să fie memorată în ordine alea-

toare. Avantajele memorării compact aleatoare constau în faptul că noi elemente nenule ale matricei pot fi adăugate la sfîrşitul zonelor de memorare fără a perturba celelalte elemente, precum şi o manevrabilitate rapidă a datelor.

In cazul matricelor simetrice această schemă de memorare se poate simplifica prin memorarea numai a elementelor de deasupra diagonalei principale, utilizînd zonele VALA, ILIN şi ICOL precum şi a elementelor diagonale într-o zonă secundară DIAG.

Numărul total de cuvinte necesar memorării unei matrice rare A (m,n) - dimensionale,(nesimetrice) prin intermediul celor trei zone VALA, ILIN şi ICOL este 3nnr. Raportul dintre cerinţele de memorie a acestei scheme şi a celei standard este C = 3r. Raportul de memorare notat r este raportul dintre numărul de elemente nenule şi numărul total de elemente.

Se observă că această schemă este eficientă pentru memorarea matricelor rare, cu o densitate de maximum 33,3%.

In Varianta 3 am descris metoda lui Fridman folosind memorarea compactă aleatoare (cu stocare în memoria centrală).

In Varianta 2 pentru metoda Fridman am utilizat tehnica descrisă la Varianta 1, cu ideile de la memorarea compactă aleatoare.

Pentru exemplificare am ales sistemul neliniar

$$Fx = 0$$

$$Fx = \begin{pmatrix} x_1 \\ \cos x_1 + x_2 \\ \vdots \\ \cos x_9 + x_{10} \end{pmatrix}$$

Să observăm că derivata Fréchet

$$F'(x) = \begin{pmatrix} 1 & & & & 0 \\ -\sin x_1 & 1 & & & \\ & -\sin x_2 & 1 & \ddots & \\ & & \ddots & \ddots & \\ 0 & & & -\sin x_9 & 1 \end{pmatrix}$$

este o matrice rară şi deci este indicată memorarea compact aleatoare. Indicii de linie, coloană, ai elementelor nenule ale lui F'(x), elementele nenule, sînt precizate în ordinea în care apar, coloană după coloană.

Prezentăm programele FORTRAN pentru Varianta 2 şi Varianta 3 astfel încît subrutinele FRID (Varianta 2) şi FRIDMAN2 (Varianta 3) să poată fi stocate într-o bibliotecă sursă.

Pentru apelarea subrutinei FRID care descrie metoda lui FRIDMAN utilizînd tehnica descrisă la Varianta 1, cu ideile de la memorarea compactă aleatoare, utilizatorul va preciza :

- Ecuaţia Fx = o prin subprogramul FUNCTION F cu structura de la Varianta 1 ;

- Elementele nenule ale derivatei Fréchet prin subprogramul FUNCTION VALA de parametri X,L,N cu X = tablou de dimensiune n, în care se reţine iteraţia curentă x^k ; cu L = numărul de ordine al elementului nenul din derivata Fréchet (luat în considerare) ; cu N = dimensiunea n a sistemului ;

- Indicii de linie ai elementelor nenule, în ordinea dată în VALA, prin programul FUNCTION ILIN(L) ;

- Indicii de coloană ai elementelor nenule, în ordinea dată în VALA, prin subprogramul FUNCTION ICOL(L).

La apelarea subrutinei FRIDMAN2, care este şi cea mai indicată pentru metoda lui Fridman cînd se lucrează cu matrici rare, se va descrie :

- Ecuaţia Fx = o prin subprogramul FUNCTION F ;

- Derivata Fréchet prin subrutina DERIV de parametri : VALA = tablou de dimensiune LD ce conţine elementele nenule ale derivatei, LD fiind numărul de elemente nenule ale derivatei Fréchet ; X = tablou de dimensiune n, în care se reţine iteraţia curentă x^k ; N = dimensiunea sistemului n ; ILIN = tablou de dimensiune LD — conţine indicii de linie ai elementelor nenule

Scanned with OKEN Scanner

ale derivatei, cu ordinea din VALA ; ICOL = tablou de dimensiune LD - conține indicii de coloană ai elementelor nenule ale derivatei, cu ordinea din VALA ;

```
C METODA LUI FRIDMAN PENTRU SISTEME NELINIARE. VARIANTA 2.
      DIMENSION X(10),PR(10)
      DATA N,EPS,LK,X,LD/10,1.E-3,50,10*0.3,19/
      CALL FRID(N,EPS,LK,LD,X,KOD,PR)
      PRINT 1,KOD
1     FORMAT(10X,'KOD=',I1)
      STOP
      END
C PRECIZAREA ECUATIEI F(X)=0.
      FUNCTION F(X,I,N)
      DIMENSION X(N)
      GOTO(1,2,3,4,5,6,7,8,9,10),I
1     F=X(1)
      RETURN
2     F=COS(X(1))+X(2)
      RETURN
3     F=COS(X(2))+X(3)
      RETURN
4     F=COS(X(3))+X(4)
      RETURN
5     F=COS(X(4))+X(5)
      RETURN
6     F=COS(X(5))+X(6)
      RETURN
7     F=COS(X(6))+X(7)
      RETURN
8     F=COS(X(7))+X(8)
      RETURN
```

Scanned with OKEN Scanner

```
9     F=COS(X(8))+X(9)
      RETURN
10    F=COS(X(9))+X(10)
      RETURN
      END
C ELEMENTELE NENULE ALE DERIVATEI FRECHET
      FUNCTION VALA(X,L,N)
      DIMENSION X(N)
      GOTO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19),L
1     VALA=1.
      RETURN
2     VALA=-SIN(X(1))
      RETURN
3     VALA=1.
      RETURN
4     VALA=-SIN(X(2))
      RETURN
5     VALA=1.
      RETURN
6     VALA=-SIN(X(3))
      RETURN
7     VALA=1
      RETURN
8     VALA=-SIN(X(4))
      RETURN
9     VALA=1
      RETURN
10    VALA=-SIN(X(5))
      RETURN
11    VALA=1
      RETURN
```

Scanned with OKEN Scanner

```
12      VALA=-SIN(X(6))
        RETURN
13      VALA=1
        RETURN
14      VALA=-SIN(X(7))
        RETURN
15      VALA=1
        RETURN
16      VALA=-SIN(X(8))
        RETURN
17      VALA=1
        RETURN
18      VALA=-SIN(X(9))
        RETURN
19      VALA=1
        RETURN
        END
C  INDICII DE LINIE AI ELEMENTELOR NENULE IN ORDINEA DATA IN VALA.
        FUNCTION ILIN(L)
        GOTO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19),L
1       ILIN=1
        RETURN
2       ILIN=2
        RETURN
3       ILIN=2
        RETURN
4       ILIN=3
        RETURN
5       ILIN=3
        RETURN
6       ILIN=4
```

Scanned with OKEN Scanner

```
      RETURN
    7 ILIN=4
      RETURN
    8 ILIN=5
      RETURN
    9 ILIN=5
      RETURN
   10 ILIN=6
      RETURN
   11 ILIN=6
      RETURN
   12 ILIN=7
      RETURN
   13 ILIN=7
      RETURN
   14 ILIN=8
      RETURN
   15 ILIN=8
      RETURN
   16 ILIN=9
      RETURN
   17 ILIN=9
      RETURN
   18 ILIN=10
      RETURN
   19 ILIN=10
      RETURN
      END
C INDICII DE COLOANA AI ELEMENTELOR NENULE DUPA ORDINEA DIN VALA.
      FUNCTION ICOL(L)
      GOTO(1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19),L
```

Scanned with OKEN Scanner

```
1       ICOL=1
        RETURN
2       ICOL=1
        RETURN
3       ICOL=2
        RETURN
4       ICOL=2
        RETURN
5       ICOL=3
        RETURN
6       ICOL=3
        RETURN
7       ICOL=4
        RETURN
8       ICOL=4
        RETURN
9       ICOL=5
        RETURN
10      ICOL=5
        RETURN
11      ICOL=6
        RETURN
12      ICOL=6
        RETURN
13      ICOL=7
        RETURN
14      ICOL=7
        RETURN
15      ICOL=8
        RETURN
16      ICOL=8
```

Scanned with OKEN Scanner

```
      RETURN
17    ICOL=9
      RETURN
18    ICOL=9
      RETURN
19    ICOL=10
      RETURN
      END
C  METODA LUI FRIDMAN
      SUBROUTINE FRID(N,EPS,LK,LD,X,KOD,PR)
      DIMENSION X(N),PR(N)
      KOD=0
      K=0
      PRINT 2
2     FORMAT(1X,'ITERATIA',10X,'SOLUTIA')
1     PRINT 3,K,(X(I),I=1,N)
3     FORMAT(1X,I2,6X,(10F12.6))
      S=0
      DO 4 I=1,N
4     S=S+F(X,I,N)*F(X,I,N)
      IF(S.LE.EPS)RETURN
      DO 5 I=1,N
5     PR(I)=0.
      DO 6 L=1,LD
      J=ICOL(L)
      I=ILIN(L)
6     PR(J)=PR(J)+VALA(X,L,N)*F(X,I,N)
      S1=0.
      DO 7 I=1,N
7     S1=S1+PR(I)*PR(I)
      IF(S1.EQ.0.) KOD=1
```

Scanned with OKEN Scanner

```
      IF(K.GE.IK) KOD=2
      IF(KOD.EQ.1.OR.KOD.EQ.2) RETURN
      K=K+1
      DO 8 I=1,N
8     X(I)=X(I)-S/SL*PR(I)
      GOTO 1
      END
```

Considerînd $x^0=(0.3,0.3,0.3,0.3,0.3,0.3,0.3,0.3,0.3,0.3)$ după 13 iteraţii am obţinut soluţia $x=(0.000000,-0.995968,-0.554246,-0.854521,-0.665221,-0.802156,-0.697052,-0785152,-0.705464,-0.769332)$.

```
C  METODA LUI FRIDMAN PENTRU SISTEME NELINIARE.MCA.VARIANTA 3.
      DIMENSION X(10),PR(10),ILIN(19),ICOL(19),VALA(19),F(10)
      DATA N,EPS,IK,X,LD/10,1.E-3,300,10*0.3,19/.
      CALL FRIDMAN2(N,EPS,IK,LD,X,KOD,PR,ILIN,ICOL,VALA,F)
      PRINT 1,KOD
1     FORMAT(10X,'KOD=',I1)
      STOP
      END
C  PRECIZAREA ECUATIEI F(X)=0
      SUBROUTINE EC(X,F,N)
      DIMENSION X(N),F(N)
      F(1)=X(1)
      DO 1 I=2,N
1     F(I)=COS(X(I-1))+X(I)
      RETURN
      END
      SUBROUTINE MODUL(F,F1,N)
      DIMENSION F(N)
      F1=0.
      DO 1 I=1,N
```

Scanned with OKEN Scanner

```
1       F1=F1+F(I)*F(I)
        RETURN
        END
C  PRECIZAREA DERIVATEI FRECHET PRIN VALA,ILIN,ICOL
        SUBROUTINE DERIV(VALA,X,LD,N,ILIN,ICOL)
        DIMENSION VALA(LD),X(N),ILIN(LD),ICOL(LD)
        L=1
        DO 3 I=1,9
        ICOL(L)=I
        ILIN(L)=I
        L=L+1
        ICOL(L)=I
        ILIN(L)=I+1
        L=L+1
3       CONTINUE
        ICOL(19)=10
        ILIN(19)=10
        DO 1 I=1,19,2
1       VALA(I)=1
        DO 2 I=2,18,2
        I1=I/2
2       VALA(I)=-SIN(X(I1))
        RETURN
        END
C  METODA LUI FRIDMAN
        SUBROUTINE FRIDMAN2(N,EPS,LK,LD,X,KOD,PR,ILIN,ICOL,VALA,F)
        DIMENSION X(N),PR(N),VALA(LD),ILIN(LD),ICOL(LD),F(N)
        KOD=0
        K=0
        PRINT 2
2       FORMAT(1X,I3,6X,(10F12.6))
```

Scanned with OKEN Scanner

```
      CALL DERIV(VALA,X,LD,N,ILIN,ICOL)
      CALL EC(X,F,N)
      CALL MODUL(F,F1,N)
      IF(F1.LE.EPS) RETURN
      DO 4 I=1,N
4     FR(I)=Ø.
      DO 5 L=1,LD
5     FR(ICOL(L))=FR(ICOL(L)+VALA(L)*F(ILIN(L))
      CALL MODUL(FR,F2,N)
      IF(F2.EQ.Ø) KOD=1
      IF(K.GE.LK) KOD=2
      IF(KOD.EQ.1.OR.KOD.EQ.2)RETURN
      K=K+1
      DO 6 I=1,N
      X(I)=X(I)-F1/F2*FR(I)
6     CONTINUE
      GOTO 1
      END
```

Am considerat acelaşi exemplu de la varianta 2, aceeaşi iteraţie iniţială şi precizie de calcul şi, după 15 iteraţii (firesc să fie astfel), am obţinut aceeaşi soluţie.

4.4. Metoda Gradientului

Pentru metoda iterativă (3.36) este prezentată o subrutină FORTRAN cu numele GRADIENT, o implementare eficientă, a metodei Gradientului pentru ecuaţii de forma $Fx = o$ cu derivata Fréchet o matrice rară. Subrutina este astfel organizată încît poate fi stocată într-o bibliotecă sursă. Pentru utilizarea ei în vederea rezolvării sistemelor neliniare se va preciza sistemul $Fx = o$ şi derivata Fréchet a lui F.

Scanned with OKEN Scanner

4.4.1. Algoritmul de calcul. Considerăm ecuaţia $Fx = o$ cu $F: D \subset \mathbb{R}^n \longrightarrow \mathbb{R}^n$, F operator neliniar şi $x^* \in D$ o soluţie a ecuaţiei. Presupunem că F este diferenţiabil Fréchet pe $S(x^*,r) \subset D$ şi matricea $F'(x)$ este mărginită pe $S(x^*,r)$ şi de rang n. Atunci (§ 3.6.3 Corolarul 1) x^* este punct de atracţie pentru iteraţia (3.36)

$$x^{k+1} = x^k - \| F'(x^k) \|^{-2} F'(x^k)^T F x^k .$$

Alegînd ca valoare de plecare $x^0 \in D$, vom obţine o soluţie aproximativă x^k a ecuaţiei, pentru care avem $\| F(x^k) \|^2 < \varepsilon$ parcurgînd următoarele etape :

Etapa 1. Se atribuie lui X - iteraţia curentă x^k, valoarea x^0;

Etapa 2. Se calculează $\| F(x^k) \|^2$. Dacă $\| F(x^k) \|^2 < EPS$ procesul de calcul se opreşte ; în caz contrar se continuă cu etapa 3 ;

Etapa 3. Se calculează $F'(x^k)^T F x^k$;

Etapa 4. Se calculează $\| F'(x^k) \|^2$;

Etapa 5. Dacă la etapa 4 se obţine zero, sau dacă se depăşeşte numărul de iteraţii (propus), procesul de calcul se opreşte, în caz contrar se continuă cu etapa 6 ;

Etapa 6. Se calculează iteraţia următoare tot în X. Se reiau operaţiile cu etapa 2.

4.4.2. Programul FORTRAN. Ca şi la metoda lui Fridman, pentru metoda (3.36), la fiecare pas de iteraţie avem nevoie de derivata Fréchet $F'(x^k) \in L(\mathbb{R}^n)$.

In § 4.3.2 am prezentat avantajele memorării compacte aleatoare pentru matricele rare. Şi pentru metoda Gradientului, am utilizat pentru stocarea derivatei Fréchet, cînd aceasta este o matrice rară, tehnica memorării compacte aleatoare utilizată la varianta 2 a metodei lui Fridman cît şi memorarea compactă aleatoare cu stocare în memoria centrală. Vom prezenta în cele ce urmează programul FORTRAN, pentru metoda Gradientului, utilizînd pentru $F'(x^k)$

matrice rară, memorarea compactă aleatoare (cu stocare în memoria centrală) fiind cea mai eficientă.

Programul poate fi folosit întocmai şi cînd $F'(x^k)$ nu este matrice rară, dar în această situaţie, memorarea compactă aleatoare nu aduce nici un avantaj în raport cu memorarea clasică.

Subrutina EC(X,F,N) calculează valoarea lui f_i pentru x^k, f_i, $i = 1,n$ fiind componentele lui F

$$Fx = \begin{pmatrix} f_1(x) \\ f_2(x) \\ \vdots \\ f_n(x) \end{pmatrix}$$

X = tablou de o dimensiune cu n elemente ce conţine iteraţia curentă x^k ;

N = dimensiunea n a sistemului ;

F = tablou de o dimensiune, cu componentele $f_i(x^k)$, $i = \overline{1,n}$;

Pentru calculul elementelor nenule ale derivatei Fréchet în x^k, folosim subrutina DERIV(VALA,X,LD,N,ILIN,ICOL). VALA = conţine elementele nenule ale derivatei Fréchet în x^k, iteraţia curentă — este un vector cu LD elemente. X = vector cu n elemente — iteraţia curentă x^k ; LD = numărul de elemente nenule ale derivatei Fréchet ; N = dimensiunea n a sistemului ; ILIN = vector cu LD elemente — conţine indicii de linie ai elementelor nenule, în ordinea din VALA ; ICOL = vector cu LD elemente — conţine indicii de coloană ai elementelor nenule, în ordinea dată în VALA ;

Observaţia 4.1. Deoarece norma de aplicaţie liniară a unei matrici corespunzătoare normei euclidiene este dificil de calculat, se poate utiliza o normă care o majorează, de exemplu $\|A\|_M = \left(\sum_{i,j=1}^{n} a_{ij}^2\right)^{1/2}$, unde a_{ij}, $i,j = \overline{1,n}$ sînt elementele matricei A.

Astfel pentru $\|F'(x^k)\|^2$ cu observaţia de mai sus şi pentru $\|F(x^k)\|^2$ vom folosi subrutina MODUL(F,F1,N) în care F = este

un vector cu n elemente, a cărui normă dorim s-o calculăm ; Fl = conţine pătratul normei vectorului F ; N = numărul de elemente ale vectorului F ;

```
C  METODA GRADIENTULUI PENTRU SISTEME NELINIARE.MCA.
       DIMENSION X(1Ø),PR(1Ø),ILIN(19),ICOL(19),VALA(19),F(1Ø)
       DATA N,EPS,LK,X,LD/10,1.E-3,300,10*0.3,19/
       CALL GRADIENT(N,EPS,LK,LD,X,KOD,PR,ILIN,ICOL,VALA,F)
       PRINT 1,KOD
1      FORMAT(1ØX,'KOD',I1)
       STOP
       END
C  PRECIZAREA ECUATIEI F(X)=Ø
       SUBROUTINE EC(X,F,N)
       DIMENSION X(N),F(N)
       F(1)=X(1)
       DO 1 I=2,N
1      F(I)=COS(X(I-1))*X(I)
       RETURN
       END
C  PRECIZAREA DERIVATEI FRECHET PRIN VALA,ILIN,ICOL
       SUBROUTINE DERIV(VALA,X,LD,N,ILIN,ICOL)
       DIMENSION VALA(LD),X(N),ILIN(LD),ICOL(LD)
       L=1
       DO 3 I=1,9
       ICOL(L)=I
       ILIN(L)=I
       L=L+1
       ICOL(L)=I
       ILIN(L)=I+1
       L=L+1
```

Scanned with OKEN Scanner

```
3     CONTINUE
      ICOL(19)=10
      ILIN(19)=10
      DO 1 I=1,19,2
1     VALA(I)=1.
      DO 2 I=2,18,2
      I1=I/2
2     VALA(I)=-SIN(X(I1))
      RETURN
      END
C  CALCULUL PATRATULUI NORMEI
      SUBROUTINE MODUL(F,F1,N)
      DIMENSION F(N)
      F1=0.
      DO 1 I=1,N
1     F1=F1+F(I)*F(I)
      RETURN
      END
C  METODA GRADIENTULUI
      SUBROUTINE GRADIENT(N,EPS,LK,LD,X,KOD,PR,ILIN,ICOL,VALA,F)
      DIMENSION X(N),PR(N),VALA(LD),ILIN(LD),ICOL(LD),F(N)
      KOD=0
      K=0
      PRINT 2
2     FORMAT(1X,'ITERATIA',10X,'SOLUTIA')
1     PRINT 3,K,(X(I),I=1,N)
3     FORMAT(1X,I5,6X,(10F12.6))
      CALL DERIV(VALA,X,LD,N,ILIN,ICOL)
      CALL EC(X,F,N)
      CALL MODUL(F,F1,N)
      IF(F1.LE.EPS)RETURN
```

```
      DO 4 I=1,N
4     PR(I)=Ø.
      DO 7 L=1,LD
      J=ICOL(L)
      I=ILIN(L)
7     PR(J)=PR(J)+VALA(L)*F(L)
      CALL MODUL(VALA,S1,LD)
      IF(S1.EQ.Ø)KOD=1
      IF(K.GE.LK) KOD=2
      IF(KOD.EQ.1.OR.KOD.EQ.2)RETURN
      K=K+1
      DO 9 I=1,N
9     X(I)=X(I)-PR(I)/S1
      GOTO 1
      END
```

În program s-a ales exemplul de la metoda lui Fridman, aceeași iterație inițială, aceeași precizie de calcul. Este cunoscut faptul că metoda Gradientului este mai lent convergentă (avînd convergență liniară) decît metoda lui Fridman. De astă dată soluția în aceleași condiții ca la Fridman, s-a obținut în 114 iterații, spre deosebire de metoda lui Fridman, cînd soluția s-a obținut în numai 15 iterații.

BIBLIOGRAFIE

1. ALTMAN,M.,(1) Connection between the method of steepest descend and Newton's method, Bull.Acad.Polon.Sci.,Ser.Sci.Math., Astronom.,Phys.5(1957),1031-1036.(2) Connection between gradient methods and Newton's method for functionals,Bull.Acad.Polon. Sci.Ser.Sci.Math.,Astronom.,Phys.6(1961),877-880. 2. ANDREI,N., RĂSTURNOIU,C.,(1) Matrice rare şi aplicaţiile lor,Ed.Tehnică, Bucureşti,1983. 3. ARGHIRIADE,E.,(1) Curs de algebră superioară vol.1,Ed.Didactică şi Pedagogică,Bucureşti,1962. 4. BELÜHINA,I., G.,(1) Asupra aproximării cu diferenţe a unor ecuaţii parabolice cu condiţii la limită,neliniare.Metode numerice şi programe, Ed.Univ.Moscova (1965),223-231. 5. BEREZIN,I.,S.,JIDKOV,N.,P., (1) Metode numerice,vol.2,Fizmatghiz,Moscova,1959.(2) Metode aproximative în analiza neliniară,Ed.Academiei RSR,Bucureşti, 1972. 6. BERNFELD,ST.,R.,LADDE,G.,S.,LAKSHMIKANTHAM,V.,(1) Existences of solutions of two points boundary value problems for nonlinear system,Univ.of Rhode Island,Technical Report nr. 35,(1975). 7. BERS,L.,(1) On mildly nonlinear partial difference equations of elliptic type,J.Res.Nat.Bur.Standards,Sect.B51 (1953),229-236. 8. BROWDER,F.,(1) Problems nonlineaires,Seminaire de mathematiques superieures,Université Montreal, 1965. 9. BUCUR,C.,M.,(1) Metode numerice,Ed.Facla,Timişoara,1973. 10. CIARLET,P.,G.,(1) Numerical analysis of the finite element method, Univ.Montreal,1976. 11. CIARLET,P.,G.,RAVIART,P.,A.,(1) General Lagrange and Hermite interpolation in R^n with applications to finite element methods,Arch.Rational Mech.Anal. 46, 3(1972),177-199. 12. CIORĂNESCU,I.,(1) Aplicaţii de dualitate în analiza funcţională neliniară,Ed.Academiei RSR,Bucureşti, 1974. 13. CIRA,O.,(1) Algoritm pentru determinarea zerourilor polinoamelor cu coeficienţi complecşi de grad superior. În : Lucrările Sesiunii Ştiinţifice a Centrului de Calcul al Univ. din Bucureşti,20-21 februarie 1987,317-324. (2) Algoritm pentru determinarea marginilor zerourilor polinoamelor din R[x] şi C[z] cu precizie impusă, Simpozionul de Matematici şi Aplicaţii,ediţia a III-a organizat de Academia RSR şi Inst.Politehnic Traian Vuia Timişoara,1989.(3) Algoritm pentru determinarea simultană a tuturor zerourilor reale ale polinoamelor din R[x], Simpozionul de Matematici şi Aplicaţii,ediţia a III-a organizată de Academia RSR şi Inst.Politehnic Traian Vuia Timişoara,1989.(4) Algorithm for determination of initial iteration of numerical methods for the solution of nonlinear equation,An. Univ.Timişoara,Ser.Sti.Mat.,vol.XXVI,fasc.3,1988,17-23. (5) Metoda cu diferenţe finite corectate pentru probleme la limită slab neliniare,Seminarul de informatică şi analiză numerică, Univ.Timişoara,1990.(6) Inversarea matricilor celular tridiagonale şi simetrice,Seminarul de informatică şi analiză numerică, Univ.Timişoara,1990. (7) Asupra metodei Lehmer-Schur pentru ecuaţii algebrice,Seminarul de informatică şi analiză numerică, Univ.Timişoara 1986. 14. COLLATZ,L.,(1) Functional analysis and numerical mathematics,trans.by H.J.Oser,New York,Academic Press, 1966. (2) Differential-Difference Equations.Applications and Numerical Problems,Workshop in Oberwolfach,W.Wetterling.Basel, Basel Birkhäuser-Verlang 1983,196 p.(International Series of Numerical Mathematics vol.62).(3) Numerical Treatment of Eigenvalue Problems,vol.3,Workshop in Oberwolfach,June,12-18,1983,Edited by Albrecht,L.Collatz,W.Welte.Basel,"Birkhäuser",1984,214 p.

Scanned with OKEN Scanner

(International Series of Numerical Mathematics,vol.69). 15.CRONIN,J.,(1) Upper an lower bounds for the number of solution of nonlinear equations,Proceed.of Symposia in Pure Math.,vol.18, part.1,A.M.S.,1970,50-61. (2) Topological degree and number of solutions of equations,Duke Math.J.38(1971),531-538. 16. CURRY, H.,(1) The method of steepest descent for nonlinear minimization proolems,Quart.Appl.Math.2(1944),258-261. 17. DEMIDOVICH, B.,P.,MARON,I.,A.,(1) Computational Mathematics,Mir Publishers, Moscow,1976. 18. DIAZ,J.,METCALF,F.,(1) On the set of subsequential limit points of succesive approximations,Trans.Amer. Math.Soc.135(1969),459-485. 19. DOUGLAS,J.,Jr.,(1) Alternating direction iteration for mildly nonlinear elliptic difference equations,Numer.Math.3(1961),92-98. 20. FORSYTHE,G.,WASOW,W., (1) Finite-difference methods for partial differential equations,John Wiley,New York,1960. 21. FRIDMAN,V.,(1) Un proces iterativ cu erori minime pentru ecuaţii operaţionale neliniare, Dokl.Akad.Nauk.SSSR,139(1961),1063-1066. (2) Despre convergenţa metodelor de tipul pantei celei mai rapide, Uspechi Mat.Nauk. 17(1962),201-204. 22. GERGEI,I.,(1) O metodă pentru rezolvarea sistemelor de ecuaţii neliniare, J.Vicisl.Mat.i Mat.Fiz.16(1976) 776-778. 23. GODUNOV,S.,K.,REABENKI,V.,S.,(1) Scheme de calcul cu diferenţe finite,Ed.Tehnică,Bucureşti,1977. 24. GUNN,J., (1) On the two-stage iterative method of Douglas for mildly nonlinear elliptic difference equations,Numer.Math.6(1964),243-249. 25. HART,H.,MOTZKIN,T.,(1) A composite Newton-Raphson gradient method for the solution of systems of equations,Pacific J.Math. 6(1965),691-707. 26. HENRICI,P.,(1) Discrete variable methods for ordinary differential equations,JohnWiley,New York, 1962. 27. KANIEL,S.,(1) Quasi-compact nonlinear operator in Banach space and applications, Arcive Rat.Mech.Anal.21(1966),259-278. 28. KANTOROVICH,L.,(1) On Newton's method for functional equations (Russian),Dokl.Akad.Nauk.SSSR 59(1948),1237-1240. (2) Functional analysis and applied mathematics (Russian), Uspechi Mat.Nauk.3(1948),89-185. (3) On Newton's method (Russian),Trudy Mat.Inst.Steklov 28(1949),104-144. 29. KELLER,H.,(1) Numerical methods for two-point boundary value problems,Ginn,Boston,1968. 30. KELLOGG,R.,(1) A nonlinear alternating direction method, Math.Comp.23(1969),23-28. 31. KERSHAW,D.,(1) The explicit inverses of two commonly occuring matrices,Math.Comp.23(1969),189-191. 32. KNUTH,D.,E.,(1) Tratat de programare a calculatoarelor. Algoritmi fundamentali,Ed.Tehnică,Bucureşti,1974.(2) Tratat de programare a calculatoarelor.Sortare şi căutare,Ed.Tehnică,Bucureşti,1976.(3) Tratat de programare a calculatoarelor.Algoritmi seminumerici,Ed.Tehnică,Bucureşti,1981. 33. KUROSH,A.,(1)Higher Algebra,Mir Publishers,Moscow,1975. 34. LEMARÉCHAL,C.,(1) Une méthode de résolution de certains systèmes nonlinéaires bien posés,C.R.Acad.Sci.Paris,Ser.A-B,272(1971),A.605-A.607. 35. LEES, M., (1) Discrete methods for nonlinear two-point boundary value problems. Numerical solution of partial differential equations, Ed.J.H.Bramble,Academic Press,New York (1966),59-72. 36. LEES, M.,SCHULTZ,M.,H.,(1) A Leray-Schauder principle for A-compact mappings and the numerical solution of two-point boundary value problems, Numerical solutions of nonlinear differential equations,Ed.D.Greenspan,John Wiley,New York (1966),167-179. 37.LEHMER,D.H.,(1) A machine method for solving polynomial equations, In: Assoc.Comput.Mach.,2,1964,151-162. 38. LEVINSON,N.,(1) The Dirichlet problem for $\Delta u = f(P)$, J.Math.Mech.12(1963),567-576. 39. MARCIUK,G.,I.,SALDUROV,V.,V.,(1) Creşterea preciziei soluţiilor în scheme cu diferenţe, Ed.Academiei RSR,Bucureşti, 1981. 40. MARINESCU,GH.,(1) Tratat de analiză funcţională I,II, Ed.Academiei RSR Bucureşti,1971,1972. (2) Analiză numerică, Ed.

Academiei RSR,Bucureşti,1974. 41. MIRUŞTER,ST.,(1) Metode numerice în rezolvarea ecuaţiilor neliniare,Ed.Tehnică,Bucureşti, 1981. (2) Metode numerice în analiza neliniară (note de curs). (3) The solution by iteration of nonlinear equations in Hilbert spaces, Proc.Amer.Math.Soc.63,1977,69-73. (4) Quasi-nonexpantivity and two classical methods for solving nonlinear equations, Proc.Amer.Math.Soc.62,1977,119-129. (5) The numerical solution of nonlinear equations, Seminarul de informatică şi analiză numerică, preprint, Univ.Timişoara,nr.2,1981. (6) An iterative method for nonlinear difference equations, An. Univ.Timişoara,Ser.Sti.Mat.14(1976),118-124. 42. MINTY,G.,(1) Monotone (nonlinear) operators in Hilbert space, Duke Math.J. 29(1962),341-346. 43. NICIOM,GH.,(1) Probleme de funcţii speciale,Ed.Didactică şi Pedagogică,Bucureşti,1988. 44. MOLDOVAN, G., (1) Scheme logice şi programe în FORTRAN,Ed.Didactică şi Pedagogică,Bucureşti,1978. 45. NASHED,M.,(1) The Convergence of the method of steepest descent for nonlinear equations with variational or quasi-variational operators,J.Math.Mech.13(1964) 765-794. (2) Steepest descent for singular linear operator equations, SIAM J. Numer.Anal.7(1970),358-362. 46. OCCORSIO,M.R., (1) O metodă iterativă pentru rezolvarea sistemelor de ecuaţii, Calcolo 4(1976),625-631. 47. ORTEGA,J.,POOLE,W.,G.,(1) Numerical methods for differential equations,Pitman Publishing Inc., 1981. 48. ORTEGA,J.,ROCKOFF,M.,(1) Nonlinear difference equations and Gauss-Seidel type iterative methods,SIAM J.Numer. Anal. 3(1966),497-513. 49. ORTEGA,J.,RHEINBOLDT,W.,(1) Iterative solution of nonlinear equations in several variables,Academic Press,New York,1970. 50. OSTROWSKI,A.,(1) Rezolvarea ecuaţiilor şi a sistemelor de ecuaţii, Ed.pentru literatură străină, Moscova,1963. 51. PASCALI,D.,(1) Operatori neliniari, Ed.Academiei RSR,Bucureşti,1974. 52. PAMAITOPOL,L.,DRĂGHICESCU,J.,C.,(1) Polinoame şi ecuaţii algebrice,Ed.Albatros,Bucureşti,1980. 53. PARTER,S.,V.,(1) Remarks on the numerical computation of solution of $\Delta u = f(P,u)$. Numerical solution of partial differential equations,Ed.J.H.Bramble,Academic Press, New York,1966,75-82. (2) Numerical methods for mildly nonlinear elliptic partial differential equations,Numer.Math.7(1965), 113-128. 54. PEACEMAN,D.,RACHFORD,H.,(1) The numerical solution of parabolic and elliptic differential equations, SIAM J. Appl.Math.(1955),28-41. 55. PETER,D.,MILLETA,G.,(1) Approximation of solutions to nonlinear equations,Numerical analysis of the finite element method. Univ.Montreal,1976. 56.PETRYSHYN, W.,V., WILLIAMSON,T.,E.,Jr.,(1) Strong and weak convergence of the sequence of succesive approximations for quasi-nonexpansive mappings,J.Math.Anal.43(1973),459-497. 57. PETRYSHYN,W., V.,(1) Direct and iterative methods for the solution of linear operator equations in Hilbert space,Trans.Amer.Math.Soc., 105(1962),136-175. (2) On a fixed point theorem for nonlinear P-compact operators in Banach space,Bull.Amer.Math.Soc., 72 (1966), 329-334. 58. POPOVICI,P.,(1) The numerical solution of the finite-difference equations for mildly nonlinear two-point boundary value problems,Seminarul de informatică şi analiză numerică,preprint,Univ.Timişoara,nr.1,1981. (2) An iterative solution of nonlinear finite-differences equations,An.Univ.Timişoara, Ser.Sti.Mat.,vol.XX,1982. (3) On Fridman's method, Seminarul de informatică şi analiză numerică,preprint, Univ.Timişoara,nr.15,1983. (4) A variant of the Gradient's method for equations $A(x)x + \phi x = 0$, Seminarul de informatică şi analiză numerică, preprint, Univ.Timişoara,nr.16,1983. (5) The Dirichlet problem for mildly nonlinear elliptic equations,preprint,

Scanned with OKEN Scanner

Univ.Timişoara,nr.17,1983. (6) On Newton's method, An.Univ.Timişoara,Ser.Sti.Mat.,vol.XXII,fasc.1-2,1984. (7) O variantă a metodei direcţiilor alternate, Colocviul naţional de "Aproximare şi optimizare", Cluj-Napoca,1984. (8) Asupra rezolvării numerice a metodei lui Fridman pentru sisteme neliniare, Seminarul de informatică şi analiză numerică, preprint, Univ.Timişoara, 1986. 59. POPOVICI,P., MĂRUŞTER,ST., (1) A simple iteration for solving the nonlinear finite-difference equations, Seminarul de informatică şi analiză numerică,preprint, Univ. Timişoara, nr.5,1981. (2) Generalized Gradient's method, An. Univ.Timişoara,Ser.Sti.Mat.,vol.XXI, fasc.1-2,1983. 60. POPOVICI,P., TOPUZU,E., (1) Asupra rezolvării numerice a unor sisteme de ecuaţii neliniare, Matematica, şcoala şi producţia, Deva,1982. 61. RUS,I.,A.,(1) Principii şi aplicaţii ale teoriei punctului fix, Ed.Dacia,Cluj-Napoca,1979. 62. ŞAMANSKII, V., (1) The method of linearization for solving nonlinear boundary value problems (Russian),Ukrain.Mat.J. 20(1968), 218-227. 63. SBURLAN,S.,(1) Gradul topologic.Lecţii asupra ecuaţiilor neliniare, Ed.Academiei RSR, Bucureşti,1974. 64.SCHECTER, S., (1) Iterations methods of nonlinear problems, Trans.Amer. Math.Soc.104, 179-189,1962. 65. ŞCHIOP,A.,I.,(1) Metode aproximative în analiza neliniară, Ed.Academiei RSR, Bucureşti, 1972. (2) Analiza unor metode de discretizare, Ed.Academiei RSR,Bucureşti, 1978. 66. SCOROBOGATKO,A., MAMUNIA,L.,(1) O metodă iterativă pentru rezolvarea ecuaţiilor de tip eliptic slab neliniare, Vîcisl.Prikl.Mat., Kiev,22(1974),68-73. 67. SIMEONOV,S.,V., (1) Rezolvarea unui sistem de ecuaţii neliniare prin aproximaţii succesive, Godisnik Inz.Stroitel Inst. 18(1966), 29-33. 68. STENGER,F., (1) An algoritm for the topological degree of a mapping in n-space, Bull.A.M.S. 81 (1975), 179-182. 69. STEPLEMAN,R.,S., (1) A caracterization of local convergence for fixed point iterations in R, SIAM J. Numer.Anal. 12(1975), 887-894. 70. STETTER,H.,J., (1) Analysis of discretization methods for ordinary differential equations (Springer Tracts in Natural Philosophy),vol.23, Springer-Verlag, Berlin-Heidelberg, New York,(1973). 71. TOMA,M., ODAGESCU,I., (1) Metode numerice şi subrutine, Ed.Tehnică, Bucureşti,1980. 72. TRICOMI,F., (1) Un teorema sulla convergenza della successioni formate della successive iterate di una funzione di una variabile reale, Giorn.Math.Battaglini, 54(1916), 1-9. 73. VAINBERG,M.,M.,(1) Metode variaţionale şi metode operatorilor monotoni, Nauka, Moscova,1972. (2) Ecuaţii neliniare cu operatori monotoni, Dokl.Acad.Nauk.188(1966), 511-513. (3) Asupra convergenţei metodei pasului descendent, Dokl.Acad.Nauk.SSSR, 129(1960),9-12. (4) Asupra convergenţei metodei pasului descendent, Sibirsk.Mat.J. 2(1961),201-220.

Scanned with OKEN Scanner

5. IARNA LA TOMIS

At[1], cum tristis Hiems sequalentia protulit ora[2]
terraque marmoreo candida facta gelu,[3]
nix iacet[1] et iactam nec sol pluviaeve resolvunt,[4]
indurat Boreas[5] perpetuamque facit.[6]
Ergo[7], ubi delicuit nondum prior[8], altera venit[9]
et solet in multis bima manere locis;[9]
tantaque commoti vis est Aquilonis[10], ut altas
aequet humo turres[11], tectaque rapta ferat.[12]
Pellibus et sutis arcent mala frigora braccis;[13]
oraque de toto corpore sola patent.[14]
Saepe sonant moti glacie pendente capilli[15]
et nitet inducto candida barba gelu.[16]

		Abl abs	
cum	protulit		T
–	iacet		
et	resolvunt	terra facta	P
–	indurat	Abl part	P
–	facit		P
ubi	delicuit		T
–	venit		P
et	solet		P
	tanta est		P
ut	aequet		CONSEC
–	ferat		CONSEC
–	arcent		P
–	patent		P
–	sonant		P
et	nitet		P

paterni – paternus, -a, -um = părintesc; adj. cu 3t., masc., G sg; atr. adj.

12. quem magnis tamen excidit ausis
(care a căzut din pricina unei mare îndrăzneli)

excidit – excido, -ĕre, -i = a cădea; vb., C III, ind. perf. a., pers III sg; PV
tamen – adv. în cele din urmă; cc t
ausis – ausum, -i = îndrăzneală, ispravă; s.c., D II, neutru, Abl pl; cc cauză
magnis – magnus, -a, -um = mare; adj. cu 3t., neutru, Abl pl; atr. adj.

13. si non tenuit (dacă nu l-a putut conduce)

tenuit – teneo, -ēre, -ui, tentum = a stăpîni, a conduce; vb., C II, ind. perf. a., pers. III sg; PV

X

①

Interim Trimorates ... legatos ad Caesarem mittunt pollicenturque sese ei dedituros atque imperata facturos et petunt ut Mandubiacium ab iniuria Cassivellauni defendat atque in civitatem mittat qui praesit imperiumque obtineat.

①. Interim Trimorates ... legatos ad Caesarem mittunt — P

interim — adv. in acest timp; ect

Trimocrates — s.p., D III, N pl; S

mittunt — mitto, ere, misi, missum = a trimite; vb., C III, ind. prez. a., pers. III pl; P V

legatos — legatus, -i = sol; s.c., D II, masc., Ac pl; e. dir.

(ad) Caesarem — Caesar, -āris; s.p., D III i; Ac sg; c. ind.

(Între timp, Trimocrații trimit soli la Cezar)

②. pollicenturque sese ei dedituros atque imperata facturos [esse] — P

pollicentur — polliceor, eri, licitus sum = a făgădui; vb. dep., C II, ind. prez. pasiv, p. III pl; P V

sese —

ei —

dedituros — dedo, -ere, didi, ditum = a preda; [esse] vb., C III; ~~part. viit. a.~~ inf. viit. a.; P V

gelu — gelus, -ūs = ger, zăpadă, s.c., D IV,
masc., Abl. sg.; c.c. cauză
inducto — induco, -ere, -xi, -ctum = a îngheța;
part. perf. p., masc., Abl. sg.;
atr. adj.

10. tantaque commoti vis est Aquilonis
(și atît de mare e forța Acvilonului declanșuit)

vis – vis, vis = forță; s.c., D III p., fem., N sg; S

tanta – tantus, -a, -um = atît de mare; adj. cu 3t., fem., N sg; n. pred.

tanta est – PN

est – sum, esse, fui; vb. cop., ind. pr. a., p. III sg

Aquilonis – Aquilo, -ōnis; s.p., D III i, masc., G sg; atr. s.g.

commoti – commotus, -a, -um = furios; adj. cu 3t., masc., G sg; atr. adj.

↓

commoveo

11. ut altas aequet humo turres
(încît egalează cu pămîntul turnurile înalte)

aequet – aequo, -āre = a egala, a netezi; vb., CI, conj. pres. a., pers. III sg; PV

turres – turris, -is = turn, palat; s.c., D III p., fem., Ac pl; c. ind.

altas – altus, -a, -um =

humo – humus, -i = pămînt; s.c., D II, fem., Abl sg; c. ind.
D sg
(pămîntului)

12. tectaque rapta ferat
(și poartă acoperișuri răpite)

ferat – fero, ferre, ~~fer~~ tuli, latum = a purta; vb., conj. pr. a., pers. III sg; PV

Scanned with OKEN Scanner

7. ergo altera venit (așadar, alta vine)
ergo - adv. - așadar
altera - alterus, -a, -um = un altul, alta;
pr. neh., fem., N sg; S
venit - venio, -ire, veni, ventum = a veni;
vb., c IV, ind. pr. a., pers. III sg; PV

8. ubi delicuit nondum prior
(când încă nu s-a topit prima)
! — nondum - adv. încă nu
delicuit - deliquesco, -ĕre, -cui = a se topi; vb., c III, ind. perf. a., pers. III sg; PV
(nix) prior - prior, prius - adj. la comp. = următoare; adj., fem., N sg; atr. adj.

9. et solet in multis hiems manere locis
(și obișnuiește să rămână în multe locuri timp de 2 ani)

solet - soleo, -ere, -itus sum = a obișnui; vb. dep., c II, ind. pr. p., pers. III sg; PV
(caut - solitur)
manere - maneo, -ēre = a rămâne; Nb., inf. pers. a.), c. dir.
locis - locus, -i = loc; N. c., N II, masc., Abl pl; cc loc
multis - multus, -a, -um = mult; adj. cu 3t, masc., Abl pl; atr. adj.
[nix] hiems - hiemus, -a, -um = de 2 ani; adj. cu 3t., fem., N sg; atr. adj.

10. tantaque commoti vis est Aquilonis
(și atît de mare e forța Acvilonului declân-
țuit)
vis – vis, vis = forță; s.c., D III p.,
fem., N sg; S
tanta – tantus, -a, -um = atît de mare; adj.
cu 3t., fem., N sg; n. pred.
tanta est – PN
est – sum, esse, fui; vb. cop., ind. pr. a., p. III sg
Aquilonis – Aquilo, -ōnis; s.p., D III i, masc.,
G sg., atr. s.g.
commoti – commotus, -a, -um = furios; adj.
cu 3t., masc., G sg., atr. adj.
↓
commoveo

11. ut altas aequet humo turres
(încît egalează cu pămîntul turnurile înalte)

aequet – aequo, -āre = a egala, a netezi;
vb., CI, conj. pres. a., pers. III sg; PV
turres – turris, -is = turn, palat; s.c.,
D III p., fem., Ac pl; c. ind.
altas – altus, -a, -um =

humo – humus, -i = pămînt; s.c., D II,
fem., Abl sg; c. ind.
D sg
(pămîntului)

12. tectaque rapta ferat
(și poartă acoperișuri răpite)
ferat – fero, ferre, ~~ferri, fa~~ tuli, latum =
= a purta; vb., conj. pr. a., pers. III sg;
PV

(de) corpore – corpus, -ōris = trup; s.c., D III i, neutru, Abl sg; c. ind.
toto – totus, -a, -um = tot; adj. cu 3t., neutru, Abl sg; atr. adj.

15 - Saepe sonant moti glacie pendente capilli (Adesea sună plitele mișcate de gheața care atîrnă)
saepe – adv. adesea
sonant – sono, -āre, -āvi, -ātum = a suna; vb., CI, ind. pr. a., pers. III pl; PV
capilli – capillus, -i = fir de păr; s.c., s.c., D II, masc., N pl; S.
moti – motus, -a, -um = pus în mișcare; adj. cu 3t., masc., N pl; atr. adj.
↓
moneo, -ere = a mișca

glacie – glacies, -ei = gheață; s.c., D V, fem., Abl sg; c. de agent
pendente – pendeo, -ēre, pependi = a atîrna; pendens, -tis = part. pr. a. ~~adj.~~; ~~pendentus~~ fem., Abl sg; (care atîrnă) atr. adj.

16. et nitet inducto candida barba gelu (și barba strălucește din pricina zăpezii înghețate) lucitoare
nitet – niteo, -ēre, -ui = a străluci; vb., CII, ind. pr. a., pers. III sg; PV
barba – barba, -ae = barbă; s.c., D I, fem., N sg; S
candida – candidus, -a, -um = alb, strălucitor; adj. cu 3t., fem., N sg; atr. adj.

tecta – tectum, -i = acoperiș; s.c., D II, neutru, Ac pl; c. dir.
rapta – raptus, -a, -um = răpit; adj. cu 3t., neutru, Ac pl; atr. adj.

13. Pellibus et sutis arcent mala frigora braccis (cu ițari cușuți și cojoace îndepărtează frigurile aspre)

arcent – arceo, -ēre, -ui = a îndepărta; vb., C II, ind. pr. a., pers. III pl; P V
frigora – frigus, -ōris = frig.; s.c., D III i, neutru, Ac pl; c. dir.
mala – malus, -a, -um = aspru, rău; adj. cu 3t., neutru, Ac pl; atr. adj.
pellibus – pellis, -is = piele, cojoc; s.c., D III p, fem., Abl pl; c. ind (mod)
sutis – ~~bra~~ sutus, -a, -um = cusut; adj. cu 3t., fem., Abl pl; atr. adj.
braccis – bracae, -ārum = ițari; s.c., D I, fem., Abl pl; cc mod

14. oraque de toto corpore sola patent (și, din tot trupul, lasă neacoperit numai obrazul)

patent – pateo, -ēre, -ui = a fi deschis; vb., C II, ind. pr. a., pers. III pl; P V.
ora – os, oris = gură, obraz; s.c., D III i, neutru, Ac pl; c. dir.
sola – solus, -a, -um = singur; adj. cu 3t., neutru, Ac pl; atr. adj.

1. cum tristis Hiems squalentia protulit ora
(când tristă iarnă și-a ridicat fața crâncenă)

hiems - hiemis, -is = zăpadă, iarnă; s.c., D III i; fem., N sg.; S

tristis - tristis, -e = trist; adj. cu 2 t., fem., N sg.; atr. adj.

ora - os, oris = față, chip; s.c., D III i, neutru, Ac pl.; c. dir.

squalentia - squaleo, -ēre = a fi aspru; vb., C II, part. pres. a. (squalens, -tis), neutru, Ac pl.; atr. adj.

protulit - profero, -ferre, -tuli, -latum = a înfățișa; vb., ind. perf. a., pers. III sg.; PV

2. terraque marmorea candida facta gelu
(și pământul, albit de zăpadă, devine strălucitor ca marmura din cauza gerului)

facta - facio, -ēre, fēci, factum = a se face, a deveni; vb., part perf. p., fem., Abl sg.;

terra facta PV

Abl abs

terra - terra, -ae; s.c., D I, fem., Abl sg.; S

terra candida facta (se face strălucitor)

Abl abs.

! candida - candidus, -a, -um = strălucitor; adj. cu 3 t., fem., Abl sg.; N - pred.

marmorea - marmoreus, -a, -um = adj. ca marmura; adj. cu 3 t., masc., Abl sg.; atr. adj. (v. gelu)

Scanned with OKEN Scanner

gelu – gelus, -us = ger; s.c., D IV, masc.,
Abl sg.; cc cauzal.
(până duine strălucitor din prima gerului ca de marmură)
3. At nix iacet (iar zăpada se așterne)
iacet – iaceo, -ēre, -ui = a se întinde; vb., c II, ind. pr. a., pers. III sg.; PV
nix – nix, nivis = zăpadă; s.c., D III i, fem., N sg.; S
4. et iactam nec sol pluviaeve resolvunt (și, așternută, nici soarele nici ploile nu o topesc)
iactam – iactus, -a, -um = așternut; adj. cu 3t., fem., Ac sg.; atr. adj. (v. nixem)
resolvunt – resolvo, -ēre, -solvi, -solutum = = a topi; vb., c III, ind. pr. a., pers. III pl; PV
sol – sol, solis = soare; s.c., D III i, masc., N sg.; S
pluviaeve – pluvia, -ae = ploaie, furtună; s.c., D I, fem., N pl.; S

5. indurat Boreas (Boreas o întărește)
Boreas – s.p., masc., N sg.; S
indurat – induro, -āre = a întări; vb., c I, ind. pr. a., pers. III sg.; PV

6. perpetuamque facit (o face perpetuă)
facit – facio, -ĕre, feci, factum = a face; vb., c III, ind. pr. a., pers. III sg; PV

perpetuam – perpetuus, -a, -um = perpetuu; adj. cu 3t., fem., Ac sg.; ~~atr.~~ cc mod (aici → adverb)

Scanned with OKEN Scanner